ISSN 0280–5316
ISRN LUTFD2/TFRT--5622--SE

Non-linear Control of a Bouncing Ball

Adam Steineck

Department of Automatic Control
Lund Institute of Technology
August 1999

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
August 1999

Document Number
ISRN LUTFD2/TFRT--5622--SE

Author(s)
Adam Steineck

Supervisor
P. Hagander
A. Astolfi (Imperial College)

Sponsoring organisation

Title and subtitle

Non-linear Control of a Bouncing Ball
(Olinjär reglering av studsande boll)

Abstract

This project is about the analysis and control of a ball bouncing on a controllable surface. The surface is modelled as a spring with a non-linear, but still continuous, characteristic causing a continuous yet swift deceleration and acceleration of the ball. The objective of the controller is to either reduce or increase the bounce-height of a ball dropped from a starting height above the surface. The motion equations of the ball are modified to generate different versions of the same basic ball-surface system: dissipation-less surface, two-dimensional and dissipative variants have been constructed using a Hamiltonian representation of the motion equations.

The model is initially tested with the surface in a fixed position, to test the surface model. It has to satisfy energy equations, Newtonian impact equations and act like a real-world surface in dynamic and equilibrium conditions. Stability criteria around equilibrium points are also evaluated using a linearized model.

Four control strategies are devised to control the bounce-height of the ball, the first being a simple approach and the last a feed-back approach. Stability criteria and equilibrium points are examined with calculus and validated by computer simulations.

The model exhibits properties in line with Newton's energy conservation laws before and after impact. No comparisons are made with test data from real impacts, but the non-linear spring model is still useful for evaluating control strategies.

The surface is assumed not to move as a result of the ball impacting it, corresponding to infinite mass of the surface. All strategies require the velocity of the ball to be measured. The controllable variable is for the first three strategies position, velocity or acceleration of the plane, whereas for the fourth the force from the plane is controlled directly to alter the stiffness of the plane. The last stratgy can in practice only be realized by mounting a jet engine on the ball to achieve direct force control.

The basic control principle for the first three is that the plane should recede while the ball is in downward travel, but for the fourth, the error between the desired and actual bounce height is used in a feed-back control law for the force from the plane. Using these strategies it is possible to effect an exponential decrease of the bounce-height, but not to stop the ball's motion completely. All simulations are carried out using the MATLAB package.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
45

Recipient's notes

Security classification

Non-linear Control of a Bouncing Ball

Adam P H Steineck
Supervisors:
Dr A. Astolfi, Imperial College
Dr P. Hagander, Lund Institute of Technology

August 18, 1999

Abstract

This project is about the analysis and control of a ball bouncing on a controllable surface. The surface is modelled as a spring with a non-linear, but still continuous, characteristic causing a continuous yet swift deceleration and acceleration of the ball. The objective of the controller is to either reduce or increase the bounce-height of a ball dropped from a starting height above the surface. The motion equations of the ball are modified to generate different versions of the same basic ball-surface system: dissipation-less surface, two-dimensional and dissipative variants have been constructed using a Hamiltonian representation of the motion equations.

The model is initially tested with the surface in a fixed position, to test the surface model. It has to satisfy energy equations, Newtonian impact equations and act like a real-world surface in dynamic and equilibrium conditions. Stability criteria around equilibrium points are also evaluated using a linearized model.

Four control strategies are devised to control the bounce-height of the ball, the first being a simple approach and the last a feed-back approach. Stability criteria and equilibrium points are examined with calculus and validated by computer simulations.

The model exhibits properties in line with Newton's energy conservation laws before and after impact. No comparisons are made with test data from real impacts, but the non-linear spring model is still useful for evaluating control strategies.

The surface is assumed not to move as a result of the ball impacting it, corresponding to infinite mass of the surface. All strategies require the velocity of the ball to be measured. The controllable variable is for the first three strategies position, velocity or acceleration of the plane, whereas for the fourth the force from the plane is controlled directly to alter the stiffness of the plane. The last stratgy can in practice only be realized by mounting a jet engine on the ball to achieve direct force control.

The basic control principle for the first three is that the plane should recede while the ball is in downward travel, but for the fourth, the error between the desired and actual bounce height is used in a feed-back control law for the force from the plane. Using these strategies it is possible to effect an exponential decrease of the bounce-height, but not to stop the ball's motion completely. All simulations are carried out using the MATLAB package.

Contents

List of Figures

Chapter 1

Introduction

A ball bouncing on a movable surface is a simple system, yet it possesses many interesting properties. It can exhibit chaotic behaviour for deterministic excitation (such as a sine wave motion of the surface) and can also be modelled as a discrete system through sampling the system at the times of impact. The standard relationship for a ball driven by gravity bouncing an a surface between incoming and outgoing velocities is:

$$v_{in}(t_k) - w_{surface}(t_k) = e(v_{out}(t_k) - w_{surface}(t_k)) \tag{1.1}$$

t_k	: Time of kth impact
t_{k+1}	: Time of the next impact
v_{in}	: The ball's velocity just before the impact
v_{out}	: The ball's velocity just after the impact
$w_{surface}$	: The velocity of the surface
e	: restitution coefficient

See Beer and Johnston [1] for an introduction to impact mechanics and Vincent [2] for discrete impacts. The ball's velocity just before an impact equals minus the velocity of the ball just after the previous impact, Equation (1.2):

$$v_{in}(t_{k+1}) = -v_{out}(t_k) \tag{1.2}$$

Assuming $e = 1$ yields:

$$v_{in}(t_{k+1}) = -v_{in}(t_k) + 2w_{surface}(t_k) \tag{1.3}$$

For a fully deterministic system, where $w_{surface}$ is known (for example periodic) and no air friction is present, t_k can be substituted for k in Equation (1.3). Discrete systems of this kind have already been extensively studied by for example Guckenheimer [3]. A nonlinear control strategy for a system where the surface has periodic motion, restitution $0 < e < 1$, can be found in Vincent [2].

Impacts have been thoroughly studied from a structural integrity point of view, see Zukas [4]. Balls bouncing against surfaces are not only encountered

in the areas of juggling or walking robots (a running animal can be modelled as a bouncing ball), but also in quantum physics, where the chaotic properties have been subject to much research effort. The ball is then a particle and the surface truly a potential energy barrier. A bouncing ball can only exhibit chaotic behaviour if the restitution coefficient is unity through the *route of doubling period*, see Luck and Mehta [5].

This project deals with continuous models and methods of controlling the bounce. The plane is modelled as a nonlinear potential energy barrier that elastically repounds the ball.

First, the system motion equations of the different versions of the ball-surface system are outlined. Dissipation-less surface, two-dimensional and dissipative variants have been constructed using the Hamiltonian representation with the distance between the ball and the plane and momentum of the ball as state variables. For a review of Hamiltonian mechanics see Kilminster [6]. The variants are examined with calculus and classical theory methods as far as possible, then the models are implemented in MATLAB.

The control strategies were based on the idea that in the real world, th method to catch a ball with an elastic surface, such as a tennis racket, is to try to match the ball's velocity and then to decelerate the surface. Hence, the plane should recede while the ball is travelling downwards, thus having a negative $w_{surface}$ in Equation (1.1).

Chapter 2

The Model

2.1 Introduction

The model was constructed to simulate a rigid ball repeatedly bouncing on a horizontal elastic[1] surface due to gravity. A real world impact is not ideal, but the ball does deform the underlying surface upon impact, its crystal lattice yields a little (in the case of a rigid ball) and eventually repulses the ball. This is the action to be emulated by the surface model.

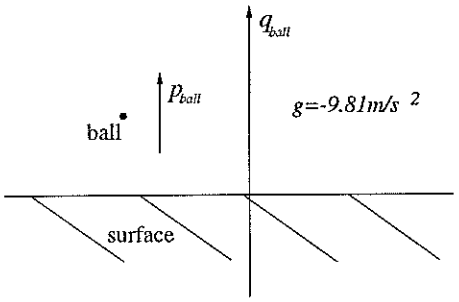

Figure 2.1: The basic system

Position of the ball is denoted by q_{ball}, momentum of the ball by p_{ball}. Positive momentum and displacement of the ball are indicated by the arrows in Figure 2.1.

The surface is modelled as a potential energy barrier in the shape of a support function, Equation (2.1). The inequality constraint of Equation (2.1) cancels spring effect on ball when ball is above the plane, $q_{ball} \geq 0$.

[1] restitution coefficient $\alpha = 1$

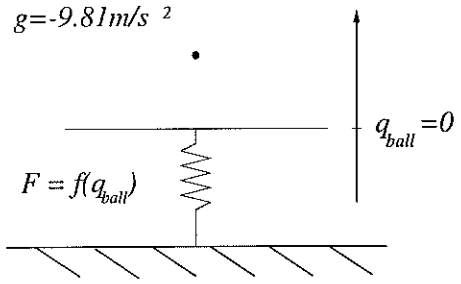

Figure 2.2: The surface as a non-linear spring

The potential energy as a function of distance for the support function is:

$$f_{sup}(q_{ball}) = \begin{cases} 0 & ; q_{ball} \geq 0 \\ e^{-\frac{1}{\lambda q_{ball}^2}} & ; q_{ball} < 0 \end{cases} \tag{2.1}$$

The advantage of using a function of this kind is that it is continuous through zero and differentiable with respect to q_{ball} for $-\infty < q_{ball} < \infty$. The potential energy plotted against displacement, q_{ball}, can be seen in Appendix A.1.1.

The surface can be thought of as a rigid plate suspended on a spring and piston. The derivative of momentum, $\dot{p}$, is force and is a function of distance only in the elastic case and is thus well represented by a spring. The dissipation, if non-zero, is a function of momentum and is represented by a piston, see Figure 2.3 for a schematic view of the plate-spring-piston-ball system. The action of piston and spring is non-linear. Their characteristics are governed by inequality constraints so the influence on the ball is analogue to encountering the potential barrier whose envelope is given by Equation (2.1).

An alternative would be to model the ball as an elastic sphere with the a potential energy function giving a rebound force as a function of radial compression of the ball. That would have prevented the use of a zero-radius ball and also been less flexible in a planar "pool-table" with multiple surfaces of nonuniform stiffness and dissipation.

2.2 The Hamiltonian System

The dissipationless system is essentially a spring with a non-linear force-distance characteristic, see Figure 2.2. No forces other than gravity affects the ball's motion when it is not in contact with the plane.

Hamiltonian systems conserve the total energy, given by an energy function $H(q_{ball}, p_{ball})$, $\dot{q}_{ball}$ is the velocity of the ball and $\dot{p}_{ball}$ is the force on the ball. Consequently, q_{ball} is the distance between the ball and plane and p_{ball} is the momentum of the ball.

The total energy is the sum of two elements: the potential and the kinetic energy. Above the plane, the potential energy is only due to gravity, inside

the surface, $q_{ball} \leq 0$ the barrier potential energy function (2.1) is added to the gravitational potential energy. The barrier function is the support function multiplied with a scalar, K_{spring}. Thus, the potential energy function is:

$$V(q_{ball}) = \begin{cases} mgq_{ball} & ; q_{ball} \geq 0 \\ mgq_{ball} + K_{spring} e^{-\frac{1}{\lambda q_{ball}^2}} & ; q_{ball} < 0 \end{cases} \tag{2.2}$$

The kinetic energy, expressed as a function of momentum of the ball, p_{ball}, is:

$$T(p_{ball}) = \frac{1}{2m} p_{ball}^2 \tag{2.3}$$

which is just another way of writing $(mv^2)/2$, the familiar expression for kinetic energy. The total energy function for the ball in Figure 2.2 is accordingly:

$$H(q_{ball}, p_{ball}) = V(q_{ball}) + T(q_{ball}, p_{ball}) \tag{2.4}$$

Using the definition of a Hamiltonian system, see Jackson [8], the motion equation representation of Equation (2.4) becomes:

$$\begin{bmatrix} \dot{q}_{ball} \\ \dot{p}_{ball} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial H}{\partial q_{ball}} \\ \frac{\partial H}{\partial p_{ball}} \end{bmatrix} =$$

$$= \begin{cases} \begin{bmatrix} \frac{p_{ball}}{m} \\ -mg \end{bmatrix} & q_{ball} \geq 0; \\ \begin{bmatrix} \frac{p_{ball}}{m} \\ -mg - \frac{K_{spring}}{\lambda} \frac{2}{2q_{ball}^3} e^{-\frac{1}{\lambda q_{ball}^2}} \end{bmatrix} & q_{ball} < 0; \end{cases} \tag{2.5}$$

2.3 The Linearized Hamiltonian System

Linearizing the Equation (2.5) around an equilibrium gives information about changes of system stability, as the ball is pulled into the plane by the force of gravity.

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial H}{\partial p} \\ -\frac{\partial H}{\partial q} \end{bmatrix} \tag{2.6}$$

A first-order linearization $\tilde{y}$, around a point x_1, x_2 of a general function of two variables $y = f_{gen}(x_1, x_2)$ is:

$$\tilde{y} = \frac{\partial f_{gen}}{\partial x_1} \tilde{x}_1 + \frac{\partial f_{gen}}{\partial x_2} \tilde{x}_2 \tag{2.7}$$

The state variables for the linearized Equation (2.9) are denoted by $\tilde{q}_{ball}$ and $\tilde{p}_{ball}$. Because the matrix in Equation (2.6) is already the first-order derivative, the linearized equation are second-order derivatives of $H(q_{ball}, p_{ball})$. The displacement and momentum of the ball at the linearization point are q_l, p_l. Equation (2.8) is the generalized expression for Equation (2.6) linearized according to the same method as Equation (2.7).

$$\begin{bmatrix} \dot{\tilde{q}}_{ball} \\ \dot{\tilde{p}}_{ball} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 H}{\partial p \partial q}(q_l, p_l) & \frac{\partial^2 H}{\partial p \partial p}(q_l, p_l) \\ -\frac{\partial^2 H}{\partial q \partial q}(q_l, p_l) & -\frac{\partial^2 H}{\partial q \partial p}(q_l, p_l) \end{bmatrix} \begin{bmatrix} \tilde{q}_{ball} \\ \tilde{p}_{ball} \end{bmatrix} \tag{2.8}$$

The expression for $\dot{q}_{ball}$ in Equation (2.5) is a function of p_{ball} only, which leads to that the derivative taken with respect to q_{ball} leaves zero. The same reasoning applies to $\dot{p}_{ball}$. The outcome is that the diagonal elements of the linearized equation are also zero. Also, for q_l, p_l to be an equilibrium point at rest, p_l has to be zero. Hence,

$$\begin{bmatrix} \dot{\tilde{q}}_{ball} \\ \dot{\tilde{p}}_{ball} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{m} \\ K_{spring} \frac{2}{\lambda} e^{-\frac{1}{\lambda q_l^2}} \left(-3q_l^{-4} + \frac{2}{\lambda} q_l^{-6}\right) & 0 \end{bmatrix} \begin{bmatrix} \tilde{q}_{ball} \\ \tilde{p}_{ball} \end{bmatrix} \tag{2.9}$$

Depending on the choice of linearization point q_l, the system changes between marginally stable to unstable, see Section 3.2.2.

2.4 The Dissipative System

With no dissipation in the system, a ball dropped from a height above the plane would bounce indefinitely, which is an unrealistic assumption about a bouncing ball. The introduction of a piston, see Figure 2.3, represents the dissipation of energy in the bounce, corresponding to a non-unity restitution coefficient in Equation (1.1). In a real surface-ball system this would correspond to the plane and ball heating up due to the impacts deforming the ball and plane.

Though the dissipative plane is a more accurate model of the plane itself, the ideal assumption of no friction against air was made throughout this project which for a test-rig would have had to been taken into account. The generalized energy function for the system shown in Figure 2.3 is:

$$H_{dis}(q_{ball}, p_{ball}) = T(q_{ball}, p_{ball}) + V(q_{ball}) + R_{gen}(p_{ball}) \tag{2.10}$$

where R_{gen} represents a generalized dissipation function. The function H_{dis} is no longer a Hamiltonian since it no longer conserves energy but the state variable representation is kept. Let the energy dissipation function $R(p)$ be:

$$R(p_{ball}) = \begin{cases} 0 & ; q_{ball} \geq 0 \\ C_{piston} \frac{p_{ball}^2}{2} & ; q_{ball} < 0 \end{cases} \tag{2.11}$$

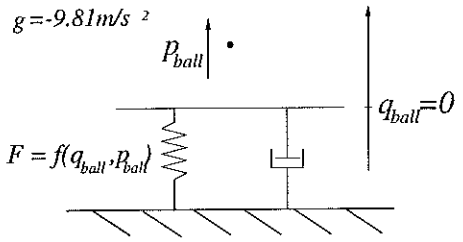

Figure 2.3: The Dissipative System

Then the derivative of $R(p_{ball})$ with respect to p_{ball} is:

$$\frac{\partial R}{\partial p_{ball}} = \begin{cases} 0 & ; q_{ball} \geq 0 \\ C_{piston} p_{ball} & ; q_{ball} < 0 \end{cases} \tag{2.12}$$

Cpiston is a scalar to change the size of the dissipation during simulations and is negative so it counteracts velocity in the motion equations.

The dissipative function also needs to be weighted with the support function, Equation (2.1), to get a smooth transistion from free-fall into the constrained motion of the dissipative surface.

$$\frac{\partial R_{tot}}{\partial p_{ball}} = \begin{cases} 0 & ; q_{ball} \geq 0 \\ C_{piston} p_{ball} e^{-\frac{1}{\lambda_c q_{ball}^2}} & ; q_{ball} < 0 \end{cases} \tag{2.13}$$

The repelling spring force and dissipation may increase differently as functions of q_{ball}, having two λ's, λ_c and λ_k respectively, allows this to be taken into consideration. The motion equations become, analoguous to the equations for the non-dissipative model of Equation (2.5):

$$\begin{bmatrix} \dot{q}_{ball} \\ \dot{p}_{ball} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial H}{\partial q} \\ \frac{\partial H}{\partial p} \end{bmatrix} =$$

$$= \begin{cases} \begin{bmatrix} \frac{p_{ball}}{m} \\ -mg \end{bmatrix} & q_{ball} \geq 0; \\ \begin{bmatrix} \frac{p_{ball}}{m} \\ -mg - \frac{K_{spring}}{\lambda} \frac{2}{q^3} e^{-\frac{1}{\lambda_k q_{ball}^2}} + C_{piston} p_{ball} e^{-\frac{1}{\lambda_c q_{ball}^2}} \end{bmatrix} & q_{ball} < 0; \end{cases} \tag{2.14}$$

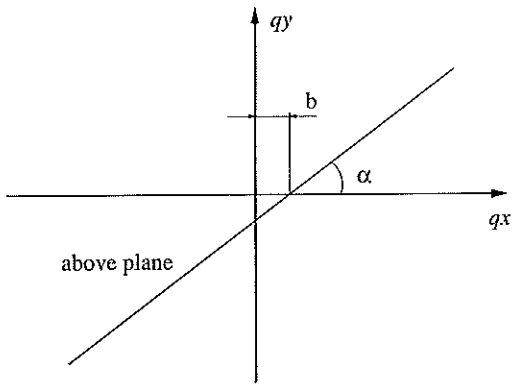

Figure 2.4: Parameters of the two-dimensional plane

2.5 The Two-Dimensional System

The one-dimensional system in Section 2.2 was expanded to movement in two dimensions, to simulate the ball lying on a flat table with the planes as vertical walls.

The plane has been described by two parameters: angle α and scalar offset b, see Figure 2.4. Two parameters were sufficient to create multiple surfaces facing different directions. This is further discussed in Section 3.3.3.

The force on the ball, $\dot{p}_{ball}$, is perpendicular to the plane, i.e. it was assumed that the ball slides frictionlessly along the surface upon impact, and its magnitude is proportional to the perpendicular distance between the ball and the surface. The perpendicular distance between the ball and the plane was calculated using the scalar product:

$$q_{\perp} = -q_{x,ball} \sin\alpha + q_{y,ball} \cos\alpha - b \tag{2.15}$$

The previous condition for the presence of an impact, $q_{ball} < 0$, has to be replaced by $q_{\perp,ball} < 0$ in the support function, Equation (2.1). Consequently, the components of the force became:

$$\begin{aligned} F_x(q_{ball}, p_{ball}) &= -F(q_{\perp,ball}) \sin\alpha \\ F_y(q_{ball}, p_{ball}) &= -mg + F(q_{\perp,ball}) \cos\alpha \end{aligned} \tag{2.16}$$

where $F(q_{ball}, p_{ball})$ may include the dissipative factor $R(p_{ball})$. This system conserves momentum for the dissipationless case, the only case in two dimensions to be simulated and tested in Section 3.2.4.

Chapter 3

Analysis of Properties

3.1 Introduction

The first part of this chapter outlines examinations of the models in Chapter 2 done with calculus, the second part outlines simulations to verify the analytic results.

All simulations have been entirely done in MATLAB 5.2 on the Imperial College Electrical and Electronic Engineering departmental system. The differential equations have been solved using the MATLAB ode23 solver. The ode23 is a one step explicit Runge-Kutta solver, that is, it computes X_{k+1} as function of X_k only[1]. The '23' stands for two or three algorithm evaluations per step.

3.2 Analytic Evaluation

3.2.1 Equilibrium for the Hamiltonian System

Knowing at which point the system in Equation (2.5) is in equilibrium is necessary so the point of linearization can be chosen. The system is in equilibrium when the momentum, p_{ball}, and force, $\dot{p}_{ball}$ equal zero. For the second condition to be true, Equation (3.1)

$$\dot{p}_{ball} = -mg - \frac{K_{spring}}{\lambda} \frac{2}{q_{ball}^3} e^{-\frac{1}{\lambda q_{ball}^2}} = 0 \tag{3.1}$$

has to be true. The expression for $\dot{p}_{ball}$ is taken from Equation (2.5).

Calculating the turning point of the ball

The Figure 3.1 depicts the potential energy well due to gravity and the surface constraining the ball's motion. The negative q-axis is "inside" the surface and the system is energy-conserving[2]. The ball accelerating from the upper turning

[1] According to the MATLAB online Help Desk or Bogacki and Shampine [9]

[2] No friction against air and elastic impact is assumed

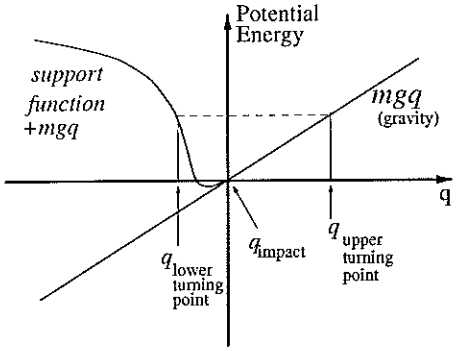

Figure 3.1: Qualitative outline of potential energy well

point towards $q = -\infty$ will upon hitting the surface convert its accumulated kinetic energy from the fall, $T(q_{impact} = 0, p_{impact}) = mgq_{upper}$ into a potential energy at the lower turning point, $V(q_{lower})$, which is due to both gravity and the surface potential. The position of the plane is assumed to be fixed. The expressions for T and V are from Equations (2.2) and (2.3).

$$mgq_{upper} = mgq_{lower} + K_{spring} e^{-\frac{1}{\lambda q_{lower}^2}} \tag{3.2}$$

Unfortunately, Equation (3.2) is not analytically solvable, because q_{lower} appears in both the mg term and in the exponent.

3.2.2 Stability of the Linearized Hamiltonian System

This section is to prove under which conditions the system is unstable, marginally stable or unstable.

Since the ball is influenced by gravity[3] the force equilibrium is at a point where the gravity downwards is equal in size to the rebound from the surface upwards.

In Figure 3.2, the mg forces of two balls of differing masses are represented by the two dashed lines, the lower the mass of the ball, the closer its line is to the x-axis. The ball is in equilibrium when its location with zero momentum coincides with an intersection between a dashed line and the solid line.

Henceforth the ball will be so light that the spring force is significantly higher than the gravity force on the ball generating a situation equivalent to only having the lower dashed line in Figure 3.2.

The region where the dashed line is under the solid line is where the force from the spring is stronger than gravity, thus the net force on the ball is upwards. Should its momentum be high enough to carry it across the force maximum, the top of the solid curve, its rate of deceleration will decrease. However, the

[3]In the case of no gravity, no equilibrium exists and the ball will always rebound eventually.

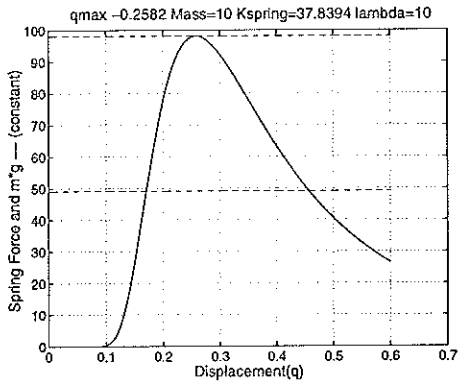

Figure 3.2: Equilibrium point locations

behaviour expected from a spring is that the force is strictly increasing as a function of displacement.

The model is thus reasonable only for displacements q_{ball} smaller than the one giving the maximum force.

Point of Maximum Opposing Force

The point of maximum was found through deriving $\dot{p}_{ball}$ in Equation (2.5) with respect to q_{ball}.

The Equation (3.3) is the expression for $\dot{p}$ in Equation (2.5).

$$\frac{d}{dq}\left(mg - K_{spring}\frac{2}{\lambda}q^{-3}e^{-\frac{1}{\lambda q^2}}\right) = K_{spring}\frac{2}{\lambda}e^{-\frac{1}{\lambda q^2}}\left(-3q^{-4} + \frac{2}{\lambda}q^{-6}\right) = 0 \tag{3.3}$$

The exponential of the above expression is never zero, so that leaves the polynomial. This gives

$$q^{-4}\left(-3 + \frac{2}{\lambda}q^{-2}\right) = 0 \Rightarrow q^2 = \frac{2}{3\lambda} \Rightarrow q = \pm\sqrt{\frac{2}{3\lambda}} \tag{3.4}$$

But the spring force is equal to zero when $q_{ball} \geq 0$ which leaves the negative solution:

$$q_{max} = -\sqrt{\frac{2}{3\lambda}} \tag{3.5}$$

Eigenvalues of the Linearized System Matrix

For the linearization point $q_l = q_{ball}$ to be an equilibrium point, mg has to be equal to the spring force and $\dot{p}_{ball} = 0$. The linearization point is thus

q_l	0	$\searrow$	$-\sqrt{\frac{2}{3\lambda}}$	$\searrow$
sign $\left(-3+\frac{2}{\lambda}q_l^{-2}\right)$	$-$	$-$	0	$+$
roots	imaginary	imaginary	$=0$	real
stability	neutral	neutral	neutral	unstable

Table 3.1: Eigenvalue sign-changes as a function of linearization point

the intersection between the dashed line and the solid line in Figure 3.2. For the intersection to move towards $-\infty$, the mass of the ball has at first to be increased until the intersection reaches the top of the solid curve of Figure 3.2. Thereafter the mass is decreased again to effect the movement of the intersection downwards, but this time the linearization point is taken to the right of the maximum of the curve.

The characteristic equation of the dissipation-less system Equation (2.9) linearized around the point q_l is:

$$\det(sI - A) = s^2 - \frac{1}{m}K_{spring}\frac{2}{\lambda}e^{-\frac{1}{\lambda q_l^2}}\left(-3q_l^{-4} + \frac{2}{\lambda}q_l^{-6}\right) \tag{3.6}$$

The roots of Equation (3.6) are thus

$$s_{0,1} = \pm\sqrt{\frac{1}{m}K_{spring}\frac{2}{\lambda}e^{-\frac{1}{\lambda q^2}}q_l^{-4}\left(-3+\frac{2}{\lambda}q_l^{-2}\right)} \tag{3.7}$$

and we get that for $q_l = -\sqrt{2/3\lambda}$ the roots become zero. In order to see for which values the system becomes unstable the sign-changes of the polynomial factor in Equation (3.6) had to be examined for $q_l \in \,]0, -\infty]$. The arrows in Table 3.1 indicates q_l strictly decreasing towards the next value in the table, in the case of the last column q_l tends towards minus infinity.

To steadily decrease q_l is equivalent to the mass of the ball continuously increasing. Also, after having passed $q_l = -\sqrt{2/3\lambda}$ it also implies that the ball has passed the point of maximum opposing force.

For $0 > q_l > -\sqrt{2/3\lambda}$ the system has got complex roots i.e. a natural frequency of:

$$\omega_{osc} = \sqrt{\frac{1}{m}K_{spring}\frac{2}{\lambda}e^{-\frac{1}{\lambda q^2}}q_l^{-4}\left(-3+\frac{2}{\lambda}q_l^{-2}\right)} \tag{3.8}$$

and that the Equation 3.7 is marginally stable for $q_l = -\sqrt{2/3\lambda}$ and unstable (one positive real root and one negative real root) for $q_l < -\sqrt{2/3\lambda}$.

3.2.3 Spring force as a function of λ

Equation (2.5) incorporates λ in two places while also being composed of a polynomial and an exponential function. The influence of λ on the interaction between the two is important as this forms the shape of the Equation (2.5).

$$\dot{p}_{ball} = -mg - K_{spring}\frac{2q^{-3}}{\lambda}e^{-\frac{1}{\lambda q_{ball}^2}} \tag{3.9}$$

Differentiating the force with respect to distance and substituting q_{ball} for $q = -\sqrt{\frac{2}{3\lambda}}$ yielded the maximum force as a function of λ:

$$F_{max} = F(q_{max}) = \frac{\partial \dot{p}}{\partial q} = K_{spring}2^{-\frac{1}{2}}3^{\frac{1}{2}}\lambda^{\frac{3}{2}}e^{-\frac{3\lambda}{4}} = const\ \lambda^{\frac{3}{2}}e^{-\frac{3\lambda}{4}} \tag{3.10}$$

The next step was to take the derivative with respective to λ. The maximum is when the derivative is equal to zero.

$$\frac{d}{d\lambda}\left(\lambda^{\frac{3}{2}}e^{-\frac{3\lambda}{4}}\right) = \lambda^{\frac{1}{2}}\left(1-\frac{\lambda}{2}\right)e^{-\frac{3\lambda}{4}} \tag{3.11}$$

The exponential and $\lambda^{\frac{1}{2}}$ factors in the right-hand side of Equation (3.11) are $\neq 0$ for all λ. Hence,

$$1-\frac{\lambda}{2} = 0 \Rightarrow \lambda = 2 \tag{3.12}$$

For a plot of the function (3.10) as a function of λ, see Appendix A.1.2.

3.2.4 Conservation of Momentum

The Hamiltonian system in Equation (2.5) without dissipation should ideally preserve momentum. Two dimensions, no gravity and a horizontal plane were assumed for the following equations similar to a ball on a pool table bouncing against a vertical wall. The onedimensional case in Equation (2.5) is a special case. The change of momentum, when the ball is not interacting with any surface, is equal to zero.

$$\dot{p}_{ball} = 0 \tag{3.13}$$

Along the vertical axis, assuming a horizontal surface, the change of momentum is given by:

$$\dot{p}_y = \begin{cases} 0 & q_y \geq 0 \\ -\frac{K_{spring}}{\lambda_k}2q_y^{-3}e^{-\frac{1}{\lambda q_y^2}} & q_y < 0 \end{cases} \tag{3.14}$$

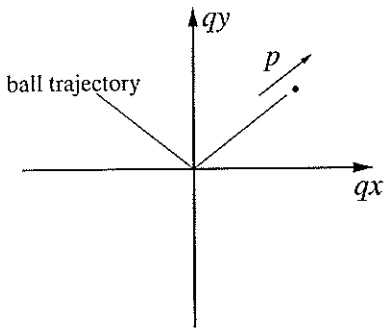

Figure 3.3: One bounce in a Two-Dimensional System

The total change of momentum along the vertical axis, can be expressed as the difference in momentum immediately after impact , t_+, minus the momentum immediately before the impact, t_-:

$$\int dp_y = \int_{t-}^{t+} -\frac{K_{spring}}{\lambda_k} 2q_y^{-3} e^{-\frac{1}{\lambda_k q_y^2}} dt \tag{3.15}$$

Using the variable substitution:

$$dt = \frac{dt}{dq}dq = \frac{1}{\dot{q}}dq = \frac{m}{p_y}dq \tag{3.16}$$

in Equation (3.15) gives Equation (3.17).

$$\int_{p_{y-}}^{p_{y+}} p_y dp_y = \int_{q-}^{q+} -K_{spring}\frac{m}{\lambda_k} 2q^{-3} e^{-\frac{1}{\lambda_k q^2}} dq \tag{3.17}$$

$$\frac{p_{y+}^2}{2} - \frac{p_{y-}^2}{2} = -K_{spring}\frac{m}{\lambda_k}\left(e^{-\left(\frac{1}{\lambda_k q_+^2}\right)} - e^{-\left(\frac{1}{\lambda_k q_-^2}\right)}\right) \tag{3.18}$$

The exponential expressions tends to zero as $q_+, q_- \to 0$, so the difference on the right hand side tends to zero as well. That is, the momentum after the impact equals the momentum prior to impact.

3.3 Analysis using Simulations

All simulations were made using MATLAB 5.2 on Unix work stations. The different models outlined in Chapter 2 will be presented in that same order in the sections Implementation followed by Simulation for each model. The ball is assumed to be a dot and in most cases its mass has been set to $m = 1$. Non-unity masses were only used to test the programmes themselves rather than model properties. That is, one simulation session was made with double the mass to check if the ball's trajectory changed when the mass of the ball changed.The simulations were conducted in order to confirm the findings in Section 3.2.

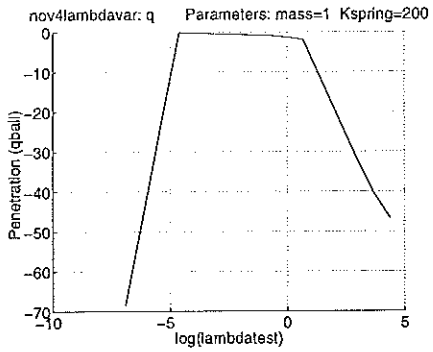

Figure 3.4: Penetration as function of λ_k

3.3.1 The One-Dimensional System Without Dissipation

Implementation

The model in Equation (2.5) was programmed into a MATLAB ode file. The spring parameters λ_k and K_{spring} had to be declared as global variables together with environmental and test parameters such as mass of ball, magnitude of gravity, if any. The inequality conditions on the spring was modelled using an `if`-statement:

```
%x=[q p] position, momentum
function xdot=nov4corrfunc(t,x)
global Kspring lambda m
    g=9.81;
    xdot=[0 ;0];

xdot(1)=x(2)/m;
if x(1)>=0
     xdot(2)=-m*g;
else
     xdot(2)=-m*g -1/lambda*Kspring*(2*x(1)^-3)*
     *exp(-1/(lambda*x(1)^2));
end
```

Effect of changing λ_k and K_{spring}

First of all the model should make sense. If the ball is too heavy or has too much momentum, it will fall through the surface. The ball does enter the surface upon impact, so one simulation to examine behaviour for different λ_k and $K_{springs}$ was carried out, see Figures 3.4 and 3.5. We know already from previous chapters that the optimum value for λ_k is 2. However, to gain a quicker

response, i.e. a steeper curve on a Force-q plot, another λ_k could be chosen but then K_{spring} has to be chosen larger. Only increasing K_{spring} without changing λ_k was far less effective to achieve a small penetration of the plane than changing both. Moreover, for very large values MATLAB gets problems with integrating the system. The penetration decreases strictly as a function of K_{spring}.

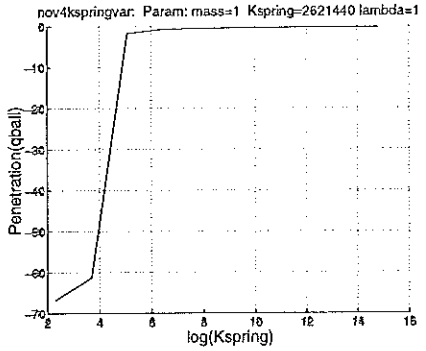

Figure 3.5: Penetration as function of K_{spring}

Energy Conservation

Energy conservation of the system was also evaluated in the simulations. The energies involved are the potential energy, kinetic energy and the energy barrier function. Simulating one bounce with the ball $q_{start} = 10$ yields the following plots:

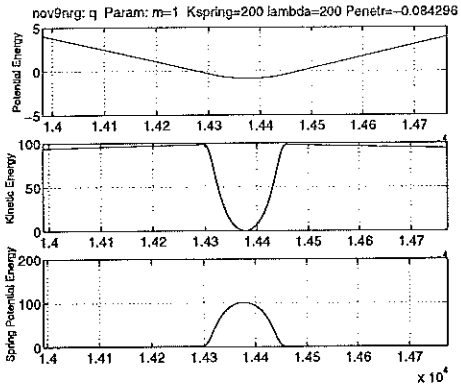

Figure 3.6: Energy conservation plot

The following program was used to examine the energy exchanges during one bounce in Figure 3.6.

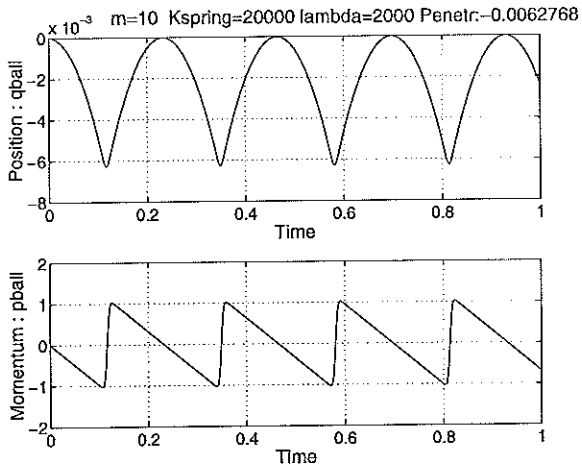

Figure 3.7: Ball position and momentum, oscillating rest

```
q=x(:,1);                    %distance
p=x(:,2);                    %momentum
Penetration=min(q);          %Finds the trajectory turning point
potentialnrg1 = m*g*q;
kineticnrg= 1/(2*m)*p .^2;
potentialnrg2=100-(potentialnrg1+kineticnrg);
```

Resting Properties

The model should also be able to keep the ball stable in the case it is just resting on the table. The figure below is a simulation of a test with the initial conditions $q_{start} = 0, p_{start} = 0$: The reason for the oscillation seen in Figure 3.7 is that exactly at $q = 0$ the only force interacting with the ball is the gravity. Also, it can be seen from Table 3.1 on page 12 that for $q < q_{maxopposingforce}$, see Section 3.2.2, the system has imaginary poles.

3.3.2 The One-Dimensional System With Dissipation

Implementation

The implementation of Equation (2.14) in MATLAB looks like this:

```
 function xdot=dissipmodel(t,x)
% VARIABLE DECLARATION
        global Kspring lambdaK lambdaC m Cpiston g
                xdot=[0 ;0];
                %For ease of reading equation
```

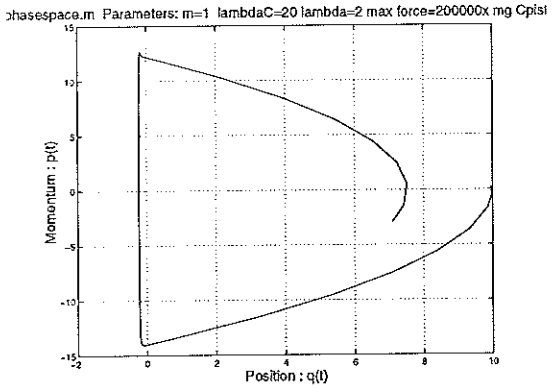

Figure 3.8: Phase space diagram of asingle bounce of Dissipative System.

```
                        q=x(1);
                        p=x(2);
                        qdot=0;
                        pdot=0;
%FUNCTION DECLARATION
qdot=p/m;
  if x(1)>=0
          pdot=-m*g;
        else
          pdot=-m*g +(-1/lambdaK*Kspring)*(2*q^-3)*exp(-1/(lambdaK*q^2))
+Cpiston*p*exp(-1/(lambdaC*q^2));
        end
```

C_{piston} is the only variable that needs to be declared. C_{piston} has to be negative, to counteract direction of travel. The lower term of `pdot` is the implementation of the piston shown in Figure 2.3 on page 7.

Simulations

The same stability cases hold true for the dissipative (or damped system) so simulations only verified that the model actually behaved as expected. See Figures 3.8 and 3.9.

3.3.3 The Two-Dimensional System Without Dissipation

Implementation of a multi-surface system

Implementing Equations (2.15) and (2.16) into MATLAB code yielded the following program. New global parameters are `alfa` and `b`. A two-dimensional problem with several surfaces meant that the system's `odefile` had to be declared in two layers. The upper level was called from the simulation file and its

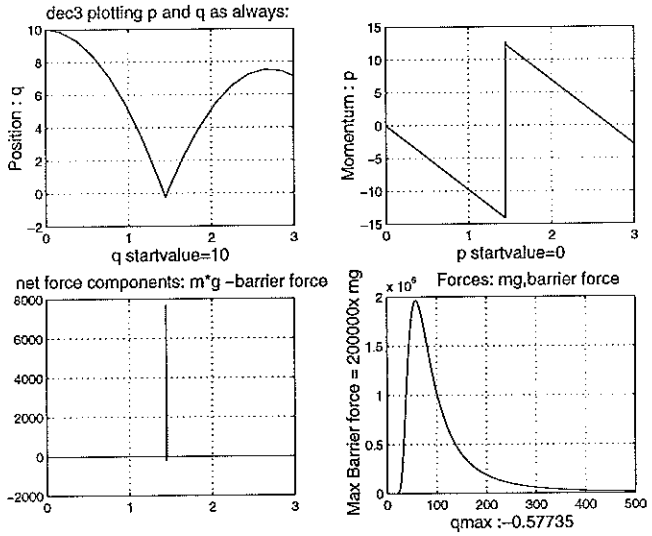

Figure 3.9: Single bounce of Dissipative System

function was to calculate the force resulting from each plane. When the ball enters the areas marked with "s" in Figure 3.10 it is only influenced by one force perpendicular to the plane, whereas in the areas marked "d" the ball is influenced by two forces, one from each plane and its amplitude decided by the perpendicular distance from the plane of influence. The separate surfaces are declared with two numbers: angle α, `alfa`[4]. in the program, and scalar product offset b (`b`). A $2 \times N$ matrix, `shape`, containing the desired specification for the N surfaces was then declared as a global variable in the main program. In the

[4]The author is aware of that the proper spelling of α is "alpha", but did not consider this while writing his MATLAB code thinking in Swedish.

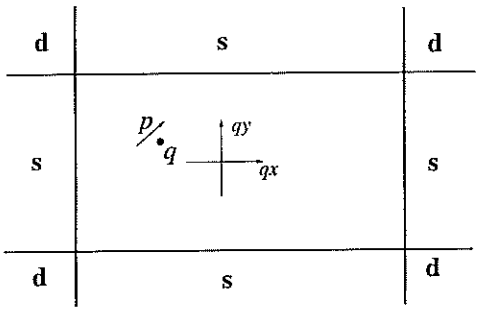

Figure 3.10: Outline of force influence areas

simulated case, $N = 4$ and

```
shape=[pi/2 -2; 0 -1; pi -1; -pi/2 -2];
```

The function `shapemodel` was the MATLAB `odefile`.

```
function xdot=shapemodel(t,x)
global shape
xdot = [0;0;0;0];

for i=1:length(shape)
        xdot = xdot + feb2model(t,x,shape(i,:));
end
```

The function `feb2model` is listed in Appendix B.2.1 and is the implementation of Equation (2.16).

Alternatively, the two-dimensional plane could have been implemented using start and end points of lines creating a polygon. This solution would have meant five parameters for each side of the polygon: one x-y pair for each end of the line, and one parameter to designate which way it is facing, moreover it was less convenient for modelling movements around intersections[5] between two planes. The areas marked "d" in Figure 3.10 on page 19 would have lacked surface potential functions, or required extra consideration.

Simulations

The purpose of the simulations was to verify that the two-dimensional model obeyed the law of Newton. Ingress angle and egress angle should be equal, which according to the lower right-hand graph of Figure 3.11 also seems to be the case with the exceptions of the two spikes which are due to using `arctan` when the ball is momentarily travelling vertically due to hitting the short end of the rectangle. The multi-surface model was only tested without gravity, the so-called pool-table case. Figure 3.11 below shows the first few bounces of one simulation. The "*" sign indicates the start of the ball's trajectory.

3.3.4 Outline of a Typical Simulation Program

The MATLAB simulations were all carried out in a similar fashion, a model implementation program qas called from a master program. The latter specified test parameters and calculated other parameters from desired properties of the function. For instance the desired final bounce-height in Section 4.3.1 is best calculated outside the implementation program, so it need not be calculated with each iteration step.

Not only test parameters were calculated in the master program but also calculations performed on the test data to display it in a desired way, be it as a

[5] two intersecting planes create a corner

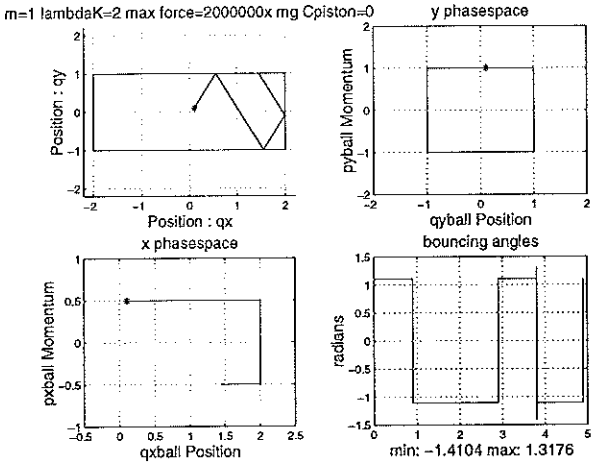

Figure 3.11: x-y graph, phase space plot, angle of travel

phase space representation or as momentum and displacement plotted against time on the x-axis.

The ode23 solver properties were also specified in the main program when calling the solver function with the implementation program file as one of the arguments for the solver.

```
options = odeset('RelTol',1e-8,'AbsTol',1e-10,'MaxStep',.1);
[t,x] = ode23('mar3model4',Tspan,xStartvalue,options);
```

Especially the parameter MaxStep was critical since a too large value would make the curved jagged rather than parabolic, and the solver might step "past" an upper turning point at which a condition might change the behaviour of the system controller. On the other hand a small value would result in longer simulation times. With four state variables, a simulation of five bounces could then take several minutes to run through. Tspan is the desired simulation time. This can either be specified as a vector containing two values, start and end point or a vector specifying every point in time at which we want the state variables calculated.

The latter method was wholly inefficient as a fixed step-length, such as below,

```
Simulationtime=[0:0.001:10]
```

would often be too large when the ball was influenced by the plane while at the same time providing unnecessary accuracy while the ball was in the trajectory part above the plane. Letting the solver adapt step length according to need but limiting the maximum length was the best method.

When plotting for instance the displacement vector, this resulted in a warped graph as the simple `plot(qvector)` command just plots all values on the y-axis.

Instead the time had to be taken into account using `plot(t,qvector)`. An entire master program can be seen in Appendix B.1.

Chapter 4

Controlling the Bounce

Controlling the impact or bounce is a problem encountered in many areas: a robot gripping an object, space craft docking to each other, or the feet of a walking robot hitting the ground. The problem of controlling the impact is difficult because the system changes dynamics rapidly, from the simple dynamics of a ball in free fall to a strictly constrained motion as it is influenced by the surface. Previous work has centred around the need to milder the strain in industrial robots at impact.

The aim of the following controllers is to control the bounce height of the ball via diminishing the resultant force on the ball through receding the plane. Thus, the controller tries to nearly match the velocity of the ball as long as the ball is influenced by the plane while travelling downwards.

Measuring the velocity of the ball, a necessity for strategies 2,3 and 4 is difficult or impractical. A possible solution would be electro-optical imaging or radar, though they are expensive.

4.1 Control Strategy 1

4.1.1 Implementation

The purpose of the controller is to avoid that the ball bounces indefinitely. From real-world experience we know that the way to catch a ball with a springy surface, such as a tennis racket, surface is to try to match the velocity of the ball and to decelerate the surface. This is what a tennis player does when catching a ball with his/her racket rather than striking the ball.

The first criterion for catching the ball by reducing the bounce is that the surface should recede when the ball impacts it. This strategy was designed to keep the plane moving downwards as long as the ball was in contact with it. When the ball leaves the surface, it resets upwards so it can move downwards as the ball is falling. The surface will describe a saw-tooth curve along the time axis.

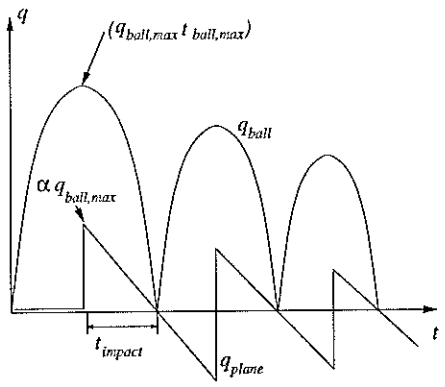

Figure 4.1: Schematic view of Strategy 1

The upper turning-point is measured, the system is deterministic, therefore time of impact can be calculated exactly and the plane set at a point and downward travel that enables it to meet the ball at precisely $q = 0$. The position and time at the upper turning-point, $q_{ball,max}, t_{max}$, are calculated according to Equation (4.1).

$$t_{impact} = \sqrt{\frac{2q_{max,ball}}{g}} \tag{4.1}$$

t_{impact} is then used to calculate a suitable downward rate of descent, which is $\dot{q}_{plane}$ according to Equation (4.2)

$$\dot{q}_{plane} = -\beta\frac{q_{max,ball}}{t_{impact}} = -\beta\sqrt{\frac{gq_{max,ball}}{2}} \tag{4.2}$$

Substituting $w_{surface} = \dot{q}_{plane}$ and because the system is deterministic , $q_{max,ball}$ completely determines v_{in} and

$$\dot{q}_{plane} = \beta\frac{v_{in}}{2} \tag{4.3}$$

in Equation(1.1), how β translates into e can be calculated.. This is equal to replacing the control action with an immobile surface ($w_{surface} = 0$) and calculating its restitution coefficient. The Equation(1.1) then becomes

$$v_{in} = -\frac{1}{(1-\beta)}v_{out} \tag{4.4}$$

and thus

$$e = \frac{1}{(1-\beta)} \tag{4.5}$$

This solution is somewhat crude but was mainly implemented to see the interaction between ball and plane, i.e. that the ball was really damped by a downward-moving plane. This proved to be the case but this strategy was not pursued beyond that because it lacked a direct coupling between energies or velocities in relation to the ball-plane system and also because of the need to know t_{impact} in advance.

4.1.2 Simulation

The program in the form of an `odefile` requires a global variable that remembers the `qplanedot` between `ode23` iterations. There is probably a more elegant solution than to use `global` variables, but this was the one that worked instantly. The new value of `qplanedot` is calculated at

$$|p_{ball}| \leq 0.1, q_{ball} > 0 \tag{4.6}$$

and is then used with the aid of a line equation to set the plane at the correct position for each iteration. The points fulfilling the criterion of Equation (4.6)

```
%Plane motion
        if and(abs(pball<=0.1),qball>0)
                        qballmax = qball;
                        qplane = qballmax*alfa;
                        tballmax = t;
                        qplanedot = -alfa*qballmax/sqrt(qballmax*2/g);

        else
                qplane = alfa*qballmax + qplanedot*(t-tballmax);
        end;
```

The above program is designed to create the movement of the plane described on Figure 4.1. The entire `function` file is listed in Section B.2.2. The vertical force from the surface is calculated as before, with the exception that the ball's absolute position, `qball`, has been substituted for `qdist = qball - qplane`.

MATLAB Simulations

The simulation showed that the program worked as predicted.

4.2 Control Strategy 2

4.2.1 Implementation

The second strategy assumes that $\dot{q}_{plane}$ is controllable. As before, catching the ball necessitates a downward movement of the surface as the ball impacts the plane. The plane should hold still after the ball has passed the lower turning point of its trajectory, and go back to zero after the ball has left the plane

as quickly as possible. The conditions for the different parts of the trajectory become:

$$\dot{q}_{plane} = \begin{cases} -k_1 q_{plane} & q_{plane} > 0, \quad \text{'reset'} \\ & \cdots \\ & q_{plane} \leq 0, \\ \alpha p_{ball} & p_{ball} \leq 0 \quad \text{'catch'} \\ 0 & p_{ball} > 0 \quad \text{'hold'} \end{cases} \tag{4.7}$$

Figure 4.3 illustrates the control action law in Equation (4.7) a displacement-momentum phase space representation. The controller decreases the egress velocity by the factor α (not to be confused with angle α of the two-dimensional system). Factor k_1 controls the speed at which the plane resets. The height of the following bounce will be decreased by the factor $(1-\alpha)^2$ since at the upper turning point, the total energy of the system is purely potential, and the total energy of the system immediately after leaving the plane is:

$$H(q_{ball}, p_{ball}, t_{impact}) = \frac{m((1-\alpha)v)^2}{2} \tag{4.8}$$

The above conditions specified in Equation (4.7) is translated into MATLAB code:

```
%Plane motion
      if qball>=0
            qplanedot=-1000*qplane; % plane resets, k1=1000
       else
            if pball<=0
                  qplanedot=alfa*pball/m;
            else
                  qplanedot=0;
            end;
       end;
```

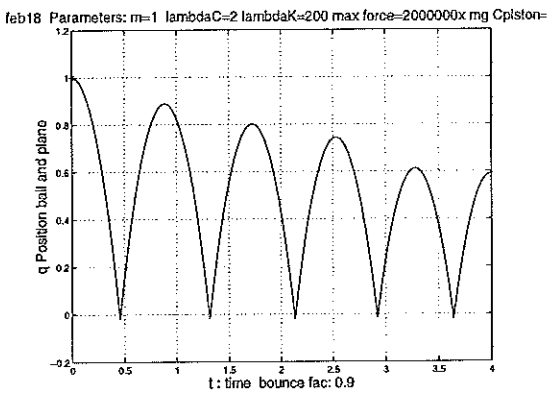

Figure 4.2: Ball trajectory for Control Strategy 1

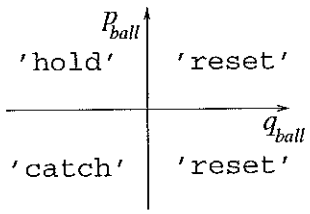

Figure 4.3: Control action for Strategy 2 for each quadrant in a p_{ball}, q_{ball} phase space

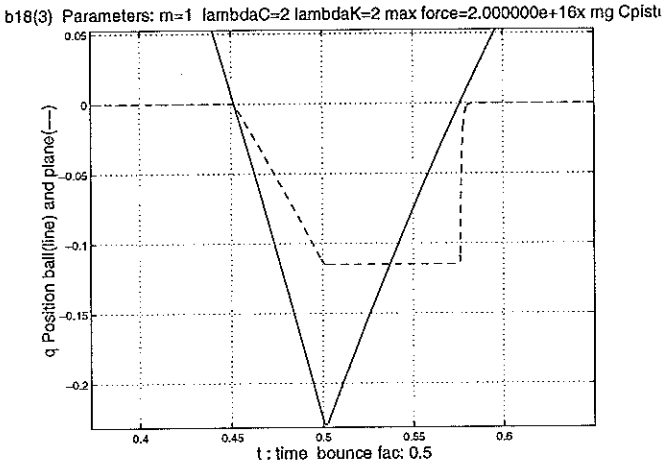

Figure 4.4: Close-up of one bounce with Control Strategy 2

The complete model implementation can be found in Section B.2.3.

4.2.2 Simulation

The figure shows the behaviour of the plane during one bounce. While the ball travels downwards, the plane follows it at a slower velocity, it holds still while the ball ascends but still in the plane and starts resetting as the ball has left the plane entirely ($q_{ball} \geq 0$). See Figure 4.4. The strategy has a drawback. When the momentum of the ball is reduced to the point at which it no longer fully leaves the plane, the plane will never start moving upwards again, and diverges towards $q_{ball} \to -\infty$. The resulting trajectory can be seen in Figure 4.5. The plane in Figure 4.5 has been made deliberately soft to better display the trajectory of the plane. The bounce occurs well inside the plane, this is due to a "lag" in the spring action as visible in Figure 3.2 on page 11.

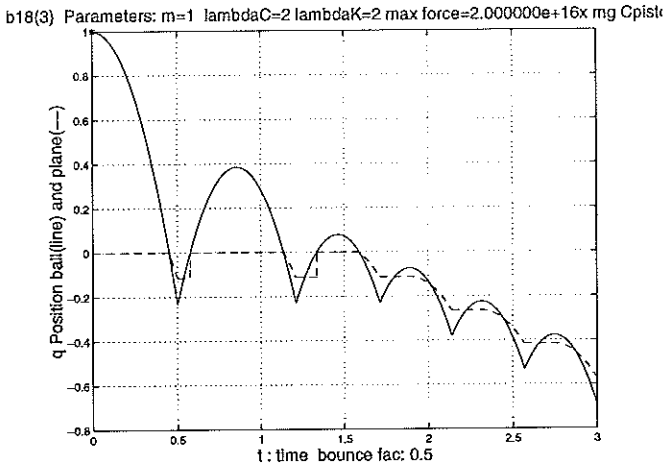

Figure 4.5: Whole plot, one bounce with Control Strategy 2

4.3 Control Strategy 3

4.3.1 Implementation

The assumption behind this controller is that $\ddot{q}_{plane}$ is the control variable, as opposed to $\dot{q}$ for Strategy 2. This demands a second-degree controller. The motion equations for the controller becomes, with the same control strategy as in Equation (4.7)

$$\ddot{q}_{plane} = \begin{cases} -k_1^2 q_{plane} - 2k_1 \dot{q}_{plane} & q_{plane} > 0, \quad \texttt{'reset'} \\ & \cdots \\ & q_{plane} \leq 0 \\ k_2 \left(\alpha \frac{p_{ball}}{m} - \dot{q}_{plane} \right) & p_{ball} \leq 0 \quad \texttt{'catch'} \\ -10^3 \dot{q}_{plane} & p_{ball} > 0 \quad \texttt{'hold'} \end{cases} \tag{4.9}$$

The entire MATLAB function can be seen in Appendix B.2.4. Equation (4.9) implemented into MATLAB code looks like this:

```
%Plane motion
        if (qball>=0|(qball <= dfac & pball<=0))
            qplanedotdot=-k1*k1*qplane-2*k1*qplanedot;
            % plane resets
        else
          if pball<=0
                qplanedotdot= (alfa*pball/m-qplanedot)*k2;
          else
```

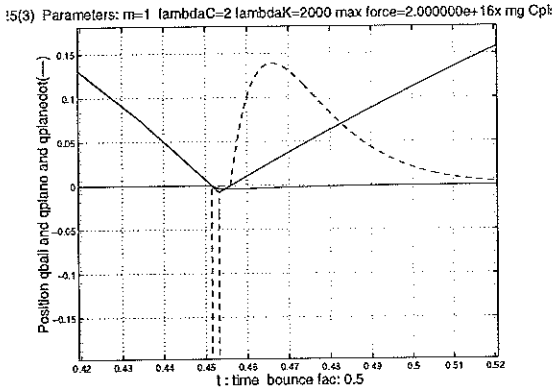

Figure 4.6: Close-up of one bounce with Control Strategy 3

```
            qplanedotdot=-1000000*qplanedot;
            end;
end;
```

The controller for the `reset` action is a second-degree critically damped controller with the double root at $s_{1,2} = k_1$. The parameter k_2 sets the speed of the controller in the `catch` mode.

4.3.2 Simulation

The controller worked as predicted. See Figure 4.6 and 4.7. The dotted line in Figure 4.6 is $\dot{q}_{plane}$. THe rapid dip in the curve thus represents a rapid acceleration downwards due to the ball hitting the plane and the controller trying to partially match the velocity of the ball. The catching speed parameter k_2 has for the shown simulation been set to 10^5 , and k_1 to 100. The latter influences how quickly the plane resets after the ball has passed $q = 0$ on its way up. As for Strategy 2, Figure 4.5, this strategy has the same problem of the controller not being able to reset due to the ball's rebound not reaching above $q_{ball} = 0$.

Remedy to Strategy 2 & 3 problem

The problem in Section 4.2.2, that the plane will diverge towards $-\infty$, can be solved for the case that the system is deterministic after leaving the plane. The upper turning point, q_{turn}, can be determined from the momentum at impact, p_{impact} if the weight of the ball is known to the controller. A momentum is translated into turning point height via Equation (4.10):

$$mgq_{turn} = \frac{1}{2m} p_{impact}^2 \Rightarrow p_{impact} = m\sqrt{2gq_{turn}} \tag{4.10}$$

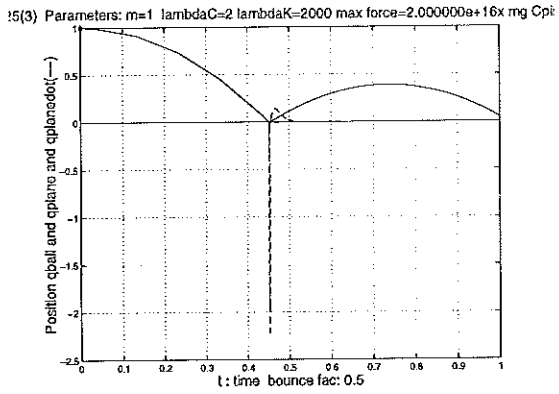

Figure 4.7: Full plot of one bounce with Control Strategy 3

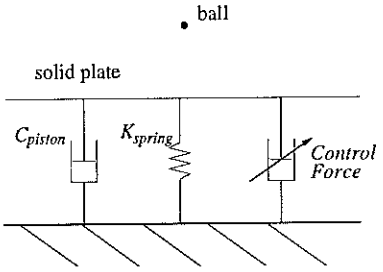

Figure 4.8: Plane with acceleration of plane controllable

The Equation (4.10) is used to create a stopping condition for the controller. The objective is to stop the system from entering the state when the ball is no longer reaching above the plane. This is accomplished by setting a condition that if the momentum at impact corresponds to a specified desired bounce height, the plane will stop draining the ball of momentum and just lie still at $q_{plane} = 0$. This was never implemented due to the fact that it is not a general solution to the problem, as it demands deterministic out-of-plane behaviour.

4.4 Control Strategy 4

4.4.1 Implementation

This strategy is designed to make the ball bounce at a preset bounce height. The control force adds a control force component to the rebound of the surface. the derivative of the force is proportional to the error in bounce-height. The present bounce-height is calculated from the momentum at impact according to Equation (4.10). Figure 4.9 is the block diagram for the force controller.

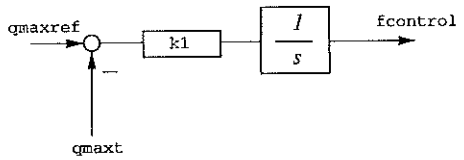

Figure 4.9: Block diagram of Strategy 4

`qmaxref`	: desired bounce height
`qmaxref`	: height of the preceding bounce
`k1`	: feedback gain variable
`fcontrol`	: control force on ball

After having reached the desired bounce, extra force input is no longer needed since the system is energy conserving, $C_{piston} = 0$. Thus if the error is zero, the control force should reset to zero as quick as possible. The MATLAB program for the intended controller looks as follows:

```
% CONTROLLER VARIABLE CALCULATION
      if (qball>0 & abs(pball)<=0.01)
              qmaxt = qball
        end;
      if abs(qmaxref-qmaxt)<.1
              fcontroldot=-100*fcontrol;
      else
              fcontroldot= k1*(qmaxref-qmaxt);
         end;
%PLANE MODEL
      if qball >= 0
              qplanedot = -k1*qplane;
              pplanedot = 0;
      else
              qplane=qball;
              qplanedot = pplane;
              pplanedot = fcontrol+(-1/lambdaK*Kspring)*(2*qplane^-3)*
                      *exp(-1/(lambdaK*qplane^2))+Cpiston*pball*
                      *exp(-1/(lambdaC*qplane^2));
      end;
%BALL MOTION
      qballdot=pball/m;
      pballdot=pplanedot-m*g;
```

The motion of the plane is designed to keep the plane 'solid' from the ball's point of view, that is the surface recedes with the ball's position if $q_{ball} < 0$. The entire program listing can be found in Section B.2.5.

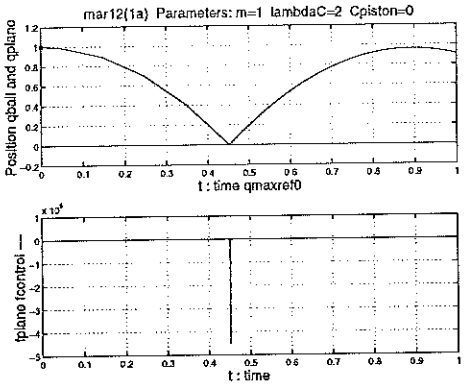

Figure 4.10: Full plot of one bounce with Strategy 4

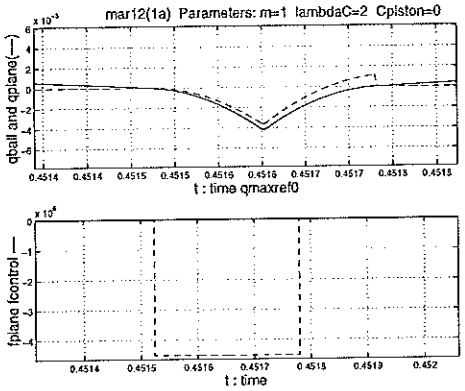

Figure 4.11: Close-up of one bounce with Strategy 4

4.4.2 Simulation

Simulating this strategy demanded too much of the memory capacity of the computer because of the 5 state variables necessary:

`qball`	: position of ball
`pball`	: momentum of plane
`qplane`	: position of plane
`pplane`	: momentum of plane
`fcontrol`	: control force

Also, the simulation time was very long, 3 hours for one bounce. The plane tries to match the position of the ball, but does not succeed entirely as the plane overshoots zero in the upswing, see Figure 4.11.

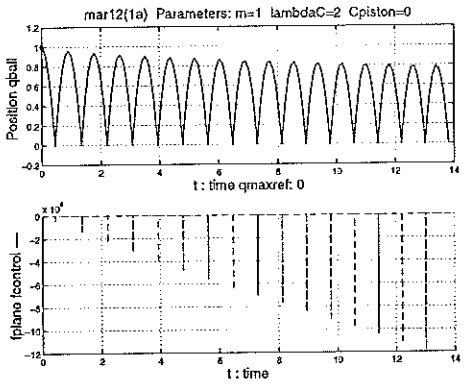

Figure 4.12: Long simulation of Strategy 4b

4.4.3 Control Strategy 4b

Because of prohibitive simulation time a simplified strategy that did not make any attempt to control the position of the plane was devised in order to eliminate two state variables, `qplane` and `pplane` as they do not directly influence the ball. The maximum simulation step length had to be kept low so the upper turning point would not be stepped past. That point is where the control force is recalculated for the next bounce.

A new problem surfaced: even though simulation went much quicker, the control force increases so rapidly that the limit of the MATLAB precision is reached. The program is unable to meet the specified integration tolerances without reducing the minimum allowable step size. See Figure 4.12 for a long simulation session.

Chapter 5

Conclusion

In this report, a model of a ball bouncing on a plane have been studied. The model has been examined using stability theory and simulations in MATLAB. The surface was modelled as a non-linear spring yielding smooth impacts. The system was shown to exhibit properties similar to those of a Newtonian, discontinuous impact.

Control strategies for controlling the bounce were also proposed and simulated, all of but one requiring the measuring of the ball's velocity before impact. The strategies employing the movement of the plane to augment or diminish the bounce are easier to implement in practice than those using the force coming from the plane to be as control variable.

A decrease of the rebound was achieved in the simulations, but not total cancellation of the bouncing motion. A real world system would not exhibit a unity restitution coefficient, so controllers taking advantage of that would the next natural step towards an algorithm for a juggling robot.

Acknowledgement

I would like to thank the following:

Dr A. Astolfi at the Control Systems Department, Imperial College London for support, help and supervision of this thesis.

Dr P. Hagander and Dr B. Wittenmark at the Control Systems Department, Lund Institute of Technology for support, advice and the opportunity to do my Master's Thesis at Imperial College.

Appendix A

MATLAB Plots

A.1 Function Envelopes

A.1.1 The Support Function

Note the orientation of the q-axis. The negative part is always assumed to be inside the surface.

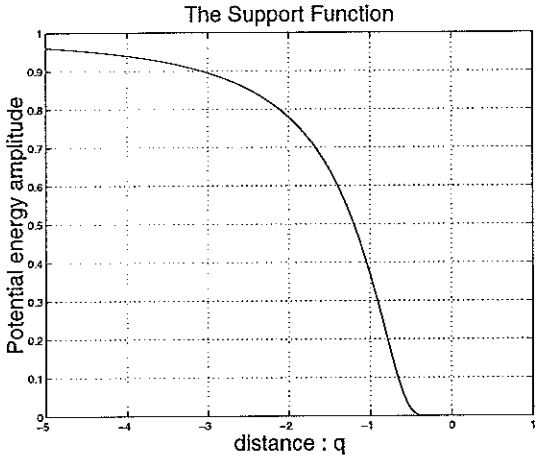

Figure A.1: The Support Function's envelope around zero

A.1.2 Envelope of Maximum Force as Function of λ

As calculated in Section 3.2.3 the force has its maximum with respect to λ at $\lambda = 2$

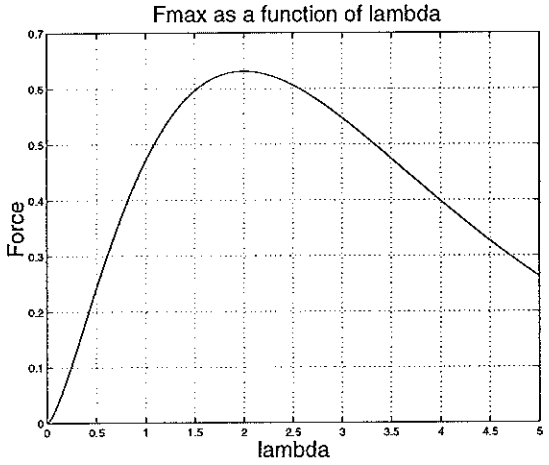

Figure A.2: Envelope of force as a function of λ

Appendix B

Program Listings

B.1 A MATLAB Master Program

This is the master program for the Control Strategy 3, Section 4.3.1. Its accompanying model implementation program can be found in Appendix B.2.4.

```
% feb25catchball3.m :
% Plane is movable, movement law incorporated in feb25model.
% See feb25a for details behind the idea.
%
% System with an energy dissipation implemented, but set to zero.
%
% alfa is elimination factor , qplane position of plane
%
clear

global Kspring m lambdaK lambdaC  g Cpiston
global alfa  k1 k2

%INIT
m=1;
g=9.81;   %
lambdaK=2000;
lambdaC=2;
Tspan = [0 1];
Cpiston   = 0;  % Cpiston should be negative
Forcefactor=2E16; % to oppose the direction of travel
xStartvalue=[1 0 0 0];
% Simulation Spec
alfa = .5;
k1  =  100;
k2 =  100000;
```

```
% Test Parameter Calculation
qmax=-sqrt(2/lambdaK/3)
Kspring=Forcefactor*m*9.81/(-1/lambdaK*(2*qmax^-3)*
*exp(-1/(lambdaK*qmax^2)));

% Simulation
options = odeset('RelTol',1e-8,'AbsTol',1e-10,'MaxStep',.1);
%options = odeset('RelTol',1e-8,'AbsTol',1e-10);
        % [t,x] = ode23('feb11model',Tspan,[xStartvalue]);
        [t,x] = ode23('feb25model3', Tspan,xStartvalue , options);

qball=x(:,1); % position
pball=x(:,2); % momentum
qplane=x(:,3); % plane position
qplanedot=x(:,4);
%GRAPHICS PRESENTATION
figure(1)
clf
%subplot(211)
plot(t,qball)
hold on
plot(t(1),qball(1),'*')
hold on
xlabel(['t : time  bounce fac: ',num2str(alfa)])
ylabel(['Position qball and qplane and qplanedot(--)'])
title(['feb25(3)  Parameters: m=',int2str(m),
'  lambdaC=',num2str(lambdaC),' lambdaK=',int2str(lambdaK),
' max force=',num2str(Forcefactor),'x mg Cpiston=',
int2str(Cpiston)]);

plot(t,qplane)
zoom on
grid on
hold on
plot(t,qplanedot,'--')

% Printfiles
% a Tspan 0..3
% alfa = .5;
% k1  =  100000;
% k2 =  100000;
%
```

B.2 Model Implementations

B.2.1 feb2model.m

This is the model implementation program for the two-dimesional model outlined in section 2.5.

```
% This one is called by shapemodel.m
% feb2model.m
% Model which includes energy dissipation as opposed to earlier versions
% alfa is inclination of plane,
% b is elevation of plane

function xdot=feb2model(t,x,shape)

% VARIABLE DECLARATION
global Kspring lambdaK lambdaC m Cpiston g alfa
xdot=[0 ;0;0;0];
%For ease of reading equation
qx=x(1); qxdot=0;
qy=x(2); qydot=0;
px=x(3); pxdot=0;
py=x(4); pydot=0;
alfa = shape(1);
b    = shape(2);
% FUNCTION DECLARATION
qxdot=px/m;
qydot=py/m;
if (-qx*sin(alfa)+qy*cos(alfa))>=b %
  % see feb2a
  pydot=-m*g;
else

%Calculation of penetration orthogonal to the plane
qortho= -qx*sin(alfa)+qy*cos(alfa);
%Calculation of forces
  pxdot= -sin(alfa)*((-1/lambdaK*Kspring)*(2*qortho^-3)*
*exp(-1/(lambdaK*(qortho^2))))
+Cpiston*px*exp(-1/(lambdaC*(qx^2)));

  pydot=-m*g + cos(alfa)*((-1/lambdaK*Kspring)*(2*qortho^-3)*
*exp(-1/(lambdaK*(qortho^2))))
+Cpiston*py*exp(-1/(lambdaC*qy^2));
end
xdot(1)=qxdot;
xdot(2)=qydot;
xdot(3)=pxdot;
```

```
xdot(4)=pydot;
```

B.2.2 feb18model.m

This is the implementation program for Control Strategy 1, Section 4.1.1.

```
% feb18model.m
% Same as feb18 but only in one dimension. On top of that this model
% needs to have a variable-position plane.
% Cpiston should be negative to oppose the direction of travel
% Position and velocity in vector as shown
% alfa is feedback constant , qplane position of plane

function xdot=feb18model(t,x)

% VARIABLE DECLARATION
global Kspring lambdaK lambdaC m Cpiston g
global alfa qplanedot qplane tballmax qplanes qballmax
xdot=[0 ;0];

%For ease of reading equation

      qball=x(1); qballdot=0;
pball=x(2); pballdot=0;

% FUNCTION DECLARATION

  %Plane motion
if and(abs(pball<=0.1),qball>0)
qballmax = qball;
qplane = qballmax*alfa;
tballmax = t;
qplanedot = -alfa*qballmax/sqrt(qballmax*2/g);

else
qplane = alfa*qballmax + qplanedot*(t-tballmax);
end;
%qplanes=[qplanes qplane];

  %Ball motion

qballdot=pball/m;

qdist=qball-qplane;
        if qdist>=0
```

```
          pballdot=-m*g;
        else
  pballdot=-m*g +(-1/lambdaK*Kspring)*(2*qdist^-3)*
*exp(-1/(lambdaK*qdist^2))+Cpiston*pball*
*exp(-1/(lambdaC*qdist^2));
        end

xdot(1)=qballdot;
xdot(2)=pballdot;
```

B.2.3 feb18model3.m

This is the implementation of the model described in section Control Strategy 2, Section 4.2.1.

```
% alfa is feedback constant , qplane position of plane
function xdot=feb18model(t,x)
% VARIABLE DECLARATION
global Kspring lambdaK lambdaC m Cpiston g
global alfa qplanedot qplane tballmax qplanes qballmax
xdot=[0 ;0];
%For ease of reading equation
qball=x(1); qballdot=0;
pball=x(2); pballdot=0;
qplane=x(3); qplanedot=0;
% FUNCTION DECLARATION

  %Plane motion
if qball>=0
qplanedot=-1000*qplane; % plane settles back
 else
if pball<=0
qplanedot=alfa*pball/m;
else
qplanedot=0;
end;
 end;

  %Ball motion

qballdot=pball/m;

qdist=qball-qplane;
        if qdist>=0
          pballdot=-m*g;
        else
```

```
  pballdot=-m*g +(-1/lambdaK*Kspring)*(2*qdist^-3)*
*exp(-1/(lambdaK*qdist^2))+Cpiston*pball*
*exp(-1/(lambdaC*qdist^2));
        end
xdot(1)=qballdot;
xdot(2)=pballdot;
xdot(3)=qplanedot;
```

B.2.4 feb25model3.m

This is the the implementation of Control Strategy 3, Section 4.3.1.

```
% acceleration of plane is controllabe
% x=[q p] postion, momentum
% Model which includes energy dissipation as opposed to earlier versions
% Cpiston should be negative to oppose the direction of travel
% alfa is feedback constant , qplane position of plane
function xdot=feb25model3(t,x)
% VARIABLE DECLARATION
global Kspring lambdaK lambdaC m Cpiston g
global  k1 k2 alfa
xdot=[0;0;0;0];
% For ease of reading equation
qball=x(1); qballdot=0;
pball=x(2); pballdot=0;
qplane=x(3);
qplanedot=x(4);
% FUNCTION DECLARATION
  %Plane motion
if qball>=0
qplanedotdot=-k1*k1*qplane-2*k1*qplanedot;
% plane settles back to zero
 else
if pball<=0
qplanedotdot= (alfa*pball/m-qplanedot)*k2;
% braking while ball is descending
else
qplanedotdot=-1000000*qplanedot;
% holding plane while ball is rebounding
end;
 end;
  %Ball motion
qballdot=pball/m;
qdist=qball-qplane;
if qdist>=0
          pballdot=-m*g;
```

```
        else
  pballdot=-m*g +(-1/lambdaK*Kspring)*
*(2*qdist^-3)*exp(-1/(lambdaK*qdist^2))+
+Cpiston*pball*exp(-1/(lambdaC*qdist^2));
        end
xdot(1)=qballdot;
xdot(2)=pballdot;
xdot(3)=qplanedot;
xdot(4)=qplanedotdot;
```

B.2.5 mar12model1.m

This is the implementation of Control Strategy 4, Section 4.4.1.

```
% Modelling the plane instead of the ball. See mar12a
% % Writing mar12model1.m
% Model which includes energy dissipation as opposed to earlier versions
% Cpiston should be negative to oppose the direction of travel
function xdot=mar12model1(t,x)
% VARIABLE DECLARATION
        global Kspring lambdaK lambdaC m Cpiston g
        global k1 qmaxref qmaxt
                xdot=[0;0;0;0;0];
                qplanedotdot=0;
                % For ease of reading equation
                        qball=x(1);     qballdot=0;
                        pball=x(2);     pballdot=0;
                        qplane=x(3);    qplanedot=0;
                        pplane=x(4);    pplanedot=0;
                        fcontrol=x(5);  fcontroldot=0;
% FUNCTION DECLARATION
  % controller variable calculation
        if (qball>0 & abs(pball)<=0.01)
                qmaxt = qball
          end;
        if abs(qmaxref-qmaxt)<.1
                fcontroldot=-100*fcontrol;
        else
                fcontroldot= k1*(qmaxref-qmaxt);
           end;
  %Plane model
        if qball >= 0
                qplanedot = -k1*qplane;
                pplanedot = 0;
        else
                qplane=qball;
```

```
                qplanedot = pplane;
                pplanedot = fcontrol+(-1/lambdaK*Kspring)*
*(2*qplane^-3)*exp(-1/(lambdaK*qplane^2))+
+Cpiston*pball*exp(-1/(lambdaC*qplane^2));
        end;
  %Ball motion
        qballdot=pball/m;
        pballdot=pplanedot-m*g;
xdot(1)=qballdot;
xdot(2)=pballdot;
xdot(3)=qplanedot;
xdot(4)=pplanedot;
xdot(5)=fcontroldot;
```

Bibliography

[1] F.P. Beer and E.R Johnston, Jr, *"Mechanics for Engineers: Dynamics"*, pp. 598-599, McGraw-Hill, Singapore, 1987.

[2] T.L. Vincent: *"Controlling a Ball to Bounce at a Fixed Height"*, in *American Control Conference* vol. 1, pp. 842-845, Seattle 1995.

[3] J. Guckenheimer and P. Holmes: *"Nonlinear Oscillations, Dynamical Systems and Bifurcations of Vector Fields"*, Springer-Verlag New York, pp. 102-116, 1983.

[4] J.A. Zukas *et al*: *"Impact Dynamics"*, John Wiley & Sons, 1982.

[5] A. Mehta and J.M. Luck: *"Bouncing Ball with a Finite Restitution: chattering, locking and chaos"*, *Physical Review E*, pp. 3988-97, vol. 48, no. 5; Nov. 1993.

[6] C.W. Kilminster *"Hamiltonian Dynamics"*, pp. 79-85, Longman's Green & Co, London, 1964.

[7] A. Astolfi *et al* : *"Proceedings of the Workshop Modelling and Control of Mechanical Systems."* Imperial College Press, pp. 219-248, 1997.

[8] A.E. Jackson *"Perspectives on Nonlinear Dynamics"*, Cambridge University Press, p236, 1989.

[9] P. Bogacki and L. F. Shampine, *"A 3(2) pair of Runge-Kutta formulas"*, Appl. Math. Letters, pp 1-9, Vol. 2, 1989.