THE UNIVERSITY OF QUEENSLAND

Bachelor of Engineering Thesis

On the Dynamics of Bells

Student Name: **Patrick Yew Cheong Chan**

Course Code: MECH4500

Supervisor:

Submission date:

A thesis submitted in partial fulfillment of the requirements of the Bachelor of Engineering degree program in Your Plan eg Minerals Process Engineering

School of Engineering

Faculty of Engineering, Physical Sciences and Architecture

Executive summary

Throughout the history of mankind, bells have been used as a communication and musical instruments. Among all the different types of bells, which were used in the modern days, the simplest form is the pendulum bells. It rotates around a pivot point, which is somewhere in the cross-section in the bell. The impact between the pendulum and the surface of the bell will create a sound. Different notes created by the bell is depend on the location where the impact between the pendulum and the wall of the bell.

For this paper, I would not be looking into the acoustic of the bell but the basic fundamental dynamic of bell. This project attempts to investigate the dynamic behaviour of the simple pendulum in both analytical and numerical method. The studies of the pendulum dynamic, impact mechanics and modal analysis are necessary for the project as it explains how pendulum bells work. Initially the free vibration is analysed to identify the relevant system parameter, subsequently this knowledge is applied to the investigation of the impact of the pendulum, which studies the clapper and the impact contact between the pendulum and the surface of the bell.

Acknowledgements

I would like thank Dr. Martin Veidt for supervising my project. His kind and thoughtful advices always were a great help for me. In addition, I would like to thank Dr. Bill Daniel and Andrew Rohde for their guidance for the development of the MATLAB GUI model and the finite analysis model.

Patrick Chan

Contents

List of Figures

List of Tables

1 Overview

1.1 Introduction

"The ringing voices of bells have comforted man in times of despair, warned him of impending danger, and accompanied him in battles, in revelry, and in worship." (Daggy 1998)

The origin of bells is traced back to the ancient times. We can imagine that when early man first stumbled onto the land, he discovered that striking objects around him such as stone, wood, shell, horn, or bone produced a pleasing sound. Not long after, a musical beat was used to highlight the ritual of tribal dances. Precisely as to where bell was first appeared is unknown; both Eastern and Western Asia claimed to be the place of origin. Whatever the case, both parties have different types of bells, which the dynamic and the acoustic are similar.

Bells were used as a communication and musical instruments for a long time. Most churches use bells as a form of communication tools for nearly half of a thousand years. Even in the present times, apart from churches, bells are also widely used in many other places such as schools, Chinese temple and small boats etc.

For centuries, bell has been used even before human kind has any knowledge of science or engineering. In the ancient day, without having any knowledge of dynamic or acoustic, we were still able to create bells and used it to communicate with one another.

The study of the bell will base on various fields of study apart from the acoustic. The vibration study is also essential to understand the mystery of the bell. It will be also very interesting to explore into other smaller details, such as the impact theorem on the bell surface of a single stroke by the clapper. Some other smaller details may have been overlooked but they do have their contributions to the bell.

This project attempts to investigate the dynamic behaviour of the clapper in both analytical and numerical methods. The studies of the pendulum dynamic, impact

mechanics and modal analysis are the fundamental findings for this project as they explain how pendulum bells work. Initially the free vibration is analysed to identify the relevant system parameter, subsequently this knowledge applied to the investigation of the impact of the pendulum, which studies the clapper and the impact contact between the pendulum and the surface of the bell.

The outline of this chapter is as follows. In the next section, I will discuss more on the motivation, which drew me to this thesis topic and followed by the description of an outline of the thesis.

1.2 Motivation

There have been many studies on the acoustic of the bell, as the single stroke by the clapper to produce the sound from the bell is crucial. Most people would prefer to investigate the acoustic in order to be able to determine how the bell sounds from different striking position on the inner surface of the bell. Numerous engineering personnel also investigated the degree of force needed to strike the bell to create the sound.

If you were going to buy a bell for your house or to change a bell in a church, the bell manufacturer would ask you which note would you prefer for the bell to sound. They would not ask you what is the length of the clapper would be. This is a common reaction for most people as how the bell sound would be a priority than the dynamic of the bell.

The public would not know that in order to improve the acoustic of the bell, ones must first know the dynamic of the bell well. The dynamic plays an important role for a traditional bell. The critical parameters such as natural frequency and the period will affect the dynamic and thus affecting how the bell sounds.

For this thesis, I would not be looking into the acoustic of the bell but the basic fundamental dynamic of bell. This thesis covers a literature review of the different types of pendulum associated with the bell and develops a simple user-friendly program. Anyone whom does not understand any computer language would be able to use the program.

1.3 Outline of thesis

The thesis is organised into five chapters. Chapter 2 aims are to provide a base understanding of how does a church bell works in the clock tower. In the subsequent chapters, I will bring you back to the study of the basic of pendulum. Chapter 3 will provide a brief understanding of how does the effect of the inner part of the bell will play a part in creating the sound.

As there is a lot of different kind sort of ringing in the world, it would be impractical to study all the different dynamic of all the different kind of bell. In fact, my focus will be on an English church bell, which will be less complex than other ringing bells.

On chapter 3, the fundamental of the pendulum would be further discussed. The profile of bell and the clapper, which is the inner of the bell, play the most crucial role in the study of the dynamic of the bell.

Upon knowing the fundamental of the pendulum using analytical method, in Chapter 4 a different approach is used to analyse the pendulum system. This method is used to compare with the result obtain in chapter 3, and to analyse the variation between the two methods.

It finished off with some suggestion for conclusion and further studies (Chapter 6).

2 The study of English church bell

2.1 Introduction

It is quite fascinating how bell makes a sound; it is a complex and interesting subject. Instead of studying the eastern profile, which has tons of different designs, we will restrict to the 'western' profile, which is much simpler and less complex. (Rayleigh 1890)

The 'western' profile only consists of a clapper, which will strike the end inner surface of the bell or hit it from outside by a hammer. Both will have the same effect and will create a sound. The impact causes the bell surface to vibrate in many different ways or modes. Every single ways will have different frequency and different time of *decaying*.

Throughout the decades, bell manufacturer have attempt to produce a 'better' shape through experimental and development (Rayleigh 1890). The sound quality or tone of the bell, ease of the casting and tuning are some of the essential aspect which the bell manufacturers would take notes. Apart from that, there are other characteristics, such as the weight and the basic dimension of the bell, are taken into consideration.

"The history of the improvement of bell sounds is one of periods of dramatic progress" (Rayleigh 1890). At other times, when the progress has slowed down or overturned, the economic and production factors will gained the upper hand. Even to this day, no one can guarantee that the profile, which they developed, produces the best result and the bell manufacturer regard their bell profile as trade secret (Simpson 1896: 150-155).

It is the purpose of this chapter to present the basic history of how do the bell works and some basic components functions of a church bell. The outline of the chapter is as follows: Chapter 2.2 will give you a brief history of a church bell and who brought the first church bell into the England's land. Chapter 2.3 provides knowledge of what is the working mechanism of a church bell, and what are the functions of the parts. Finally, Chapter 2.4 will describe briefly how a church bell makes their sound.

2.2 History of church bell

On the early days, the casting of bronze bell was not available in Europe. Only hammered church bell was available for churches in Europe. A reason suggested was casting was not permitted in religious grounds as the statues and the idols of pagan Rome were made of bronze (Daggy 1998).

Only at the seventh century, bronze bell was first appeared in England and it was from Italy by Benedict Biscop. It was used to call for prayer, toll for any departed and the awakening (Daggy 1998). Cast bronze bells made their appearance only on a limited scale in Western Europe in the same century. The development of the cast bell marked the beginning of the evolution of various types of modern bells (Rayleigh 1890).

Before that, there is no record on any circular cast bell seen during the first century when Christian church is expanding (Simpson 1896: 150-155). At that time, two distinctive bells were developed: a cup-shaped shallow bell which have no clapper and it needed to be struck from outside with a hammer to produce a sound and the one with deeper conical shape with uniform wall thickness and a clapper in it (Simpson 1896: 150-155).

Of the two different bells, the one with a clapper was used in most church tower as the longer the length of the pendulum, the angle of swing would be better to suit the clapper (Daggy 1998). From the eight to the fifteenth centuries, there were continuous changes made to the bell. Most of the manufacturers try to improve the tone quality of the bell (Daggy 1998).

As time goes on, bells underwent a lot of modification, for example the wall of the bell was changed from a convex to a straight and finally to a concave shape. For a bigger bell, it was discovered that if a reinforcement ring were placed at the bottom of the bell, it would prevent the bell from cracking. Another added advantage for the ring was that it would produce a deeper and musical tone. This rim is now known as the sound bow, a thicken wall just above the edge of a bell (Rayleigh 1890).

Only at the end of the fourteenth century, it marked the turning point of bell. There was intense study of acoustic where a truer pitch and clearer tone were further investigated; the efforts to making a ‘perfect’ bell become popular and widespread (Daggy 1998). In conclusion, a bell must contain three octaves and a minor third above the fundamental (Rossing 1991: 2196-2198). With this knowledge, people are trying to manufacture a near to perfect bell.

Compared to the bells of the sixteenth century or even today, the tone of the early church bells is considered average. Now it is made possible for bell that hangs on the tower top to weigh several tons. Without the effort from the founder, we would not have a more musical bell and the carillon would not be present (Daggy 1998).

2.3 Component of a bell

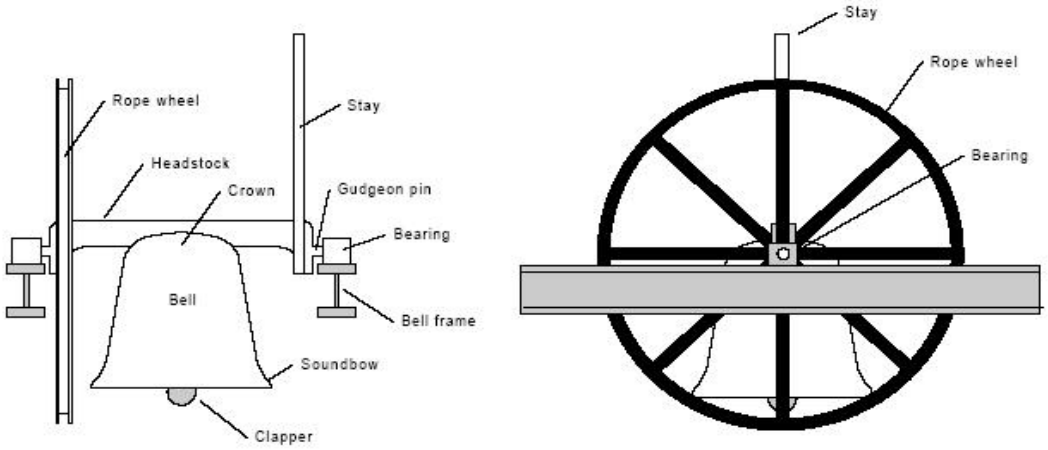


Figure 2-1: Schematic diagram of church bell (source: Blakeborough 2001: 69-92)

The above is a figure of a typical church bell. Apart from the material and the technology used for the bell, most of the arrangement is similar to the one used in the seventeen century (Blakeborough 2001: 69-92).

The crown is bolted onto the headstock and is pivoted on the gudgeon pin in a bearing, which is mounted on the bell frame. With the modern technology, the bell frame and the headstock is being replaced with iron instead of wood, which will provide a better support and will take a longer time to fail. The bearings used were also been replaced with ball bearing instead of just plain bearing. For the clapper it is designed to strike the sound bow to produce the sound. It is hanged inside of the bell, which is pivoted at the crown but is slightly below the gudgeon pin. By pulling the rope, which is tied to the pulley, the rope wheel, which is set in the headstock from below, will set the bell to move, and thus making the sound from the bell.

2.4 How does church bell makes their sound

Before pulling the rope to make the bell swings, it is moves from the ‘down’ position to ‘set’ position. The ‘down’ position is known as the clapper is in the centred of the bell frame and the bell is in equilibrium. While the ‘set’ position is to swing the bell up to an inverted position with the bell mouth pointing upwards and most of the rope is coiled around the pulley, this is called the forehand position (Hibbert 2004). The clapper will be resting on the other side of bell frame beyond the vertical axis. To be able to maintain in this position, the bell is maintained by the stay and is cantilevered by the headstock (Terhardt 2000).

When the bell rope is pulled, the bell will swing over the equilibrium position. As the bell begins to swing, the clapper will first carry the bell (The Sound and Tuning of Church Bells). As the bell swings down, the clapper will separate from the bell, swings at a greater rate, and strike the sound bow of the bell. At the top of the stroke, the stay will reengaged with the slider and the bell is left in a backhand position (Rayleigh 1890). After completing, the clapper will almost immediately return to its ‘set’ position and the pulley will be back to the forehand position (Blakeborough 2001: 69-92).

It is a good practice to left the bell in the ‘down’ position after used. However sometimes the bell would be left in the ‘set’ position, although it is not a very good practice but it will be more convenience for the ringer as he will not arrived early to setup for service (Hibbert 2004).

3 The fundamental of church bell

3.1.1 Introduction

In order to make a single note from the bell, there are a large number of working mechanisms, which we needed to be considered. To produce a clear beat, the striking position is designed to be located at the centre of the bell's height (Rossing 1991: 2196-2198). Therefore, the striking position becomes a critical fundamental for a good and clear beat from the bell (Lee, Kim, Lee, Jeong and Choi 2002: 779-790).

The fundamental of the church bell will include:

i. The shape profile of the bell, as it is mentioned in the previous chapter. The past founder tried to investigate the church bell in order to produce a bell, which can produce a perfect bell. Unfortunately, most of the documentation maybe kept within the manufacturer as trade secret. It is then say that the profile would be also an important issue for me to look into. As the profile of the bell is also a part of an element in the study of the dynamic.

ii. The study of the clapper is also an important part in investigating the whole dynamic of a church bell. There are two pivot points in the system as the equation of the bell is non-linear. It will be quite complex when there are two different components swinging at different velocity and different angles of swings. Apart from the angles and velocity, the masses and the length of the pendulum will be different as well.

    For a real life model of the church bell, it is quite impossible for the clapper in the bell to have an angle of swing less than 20 degrees and yet creating a sound. Therefore in the model, it must be assumed that the angle of swing for the clapper and the pendulum is large (cannot assumed that $\sin\theta \neq \theta$).

In section 3.2, I will be investigating the profile of the bell. The centre of mass of the bell will be needed to determine the location of the natural frequency of the bell. In section 3.3, a model of a simple pendulum with a given length and mass will be investigated. This is to provide a basic understanding of the fundamental affecting the swinging pendulum. In addition, the model will help to predict the frequency and the period of the pendulum. Different types of pendulums relating to the church bell would also be described in this section.

3.2 The profile of a church bell

It would be difficult for one to determine the centre of mass for a cylindrical form of shape using simple mathematical methods. Using a 2-dimension model will not be able to predict the centre of mass accurately; therefore using a 3-dimension model is essential to predict the centre of mass for the bell.

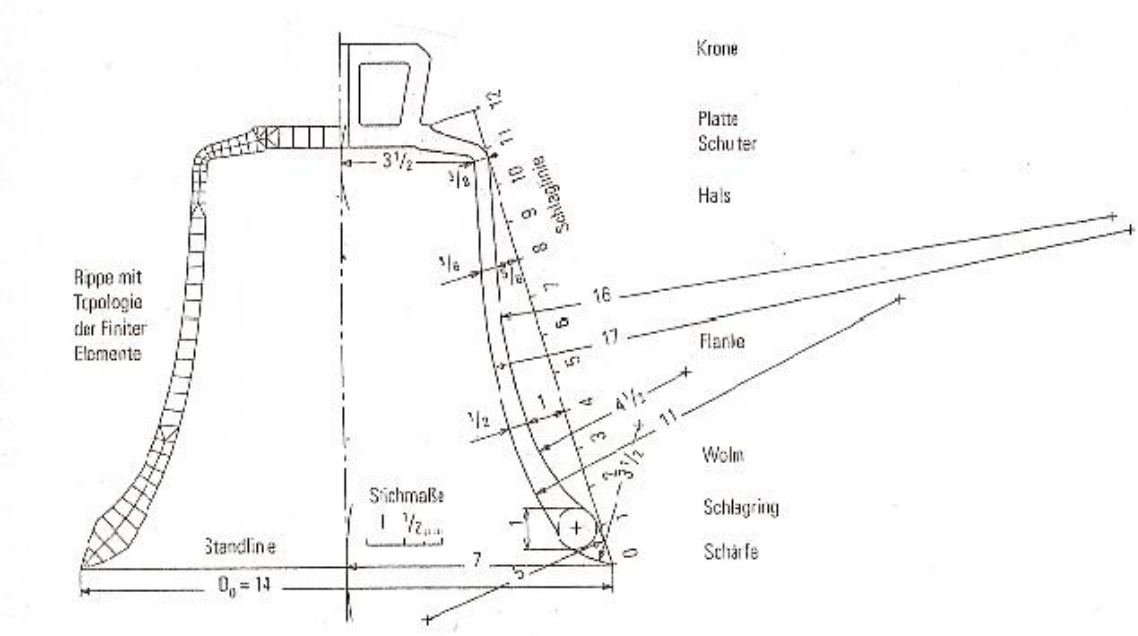


Figure 3-1: Dimension of typical church bell

The above figure shows a basic dimension of a typical church bell. The dimension is in a form of ratio (e.g. if assumed 1cm = 1 unit). The shape of a typical church bell will be developed, most of the manufacturer will follow this blueprint to develop church bell.
Using the above figure, I have incorporated the dimensions on the drawing into finite element software (STRAND 7) to locate the centre of mass.

.

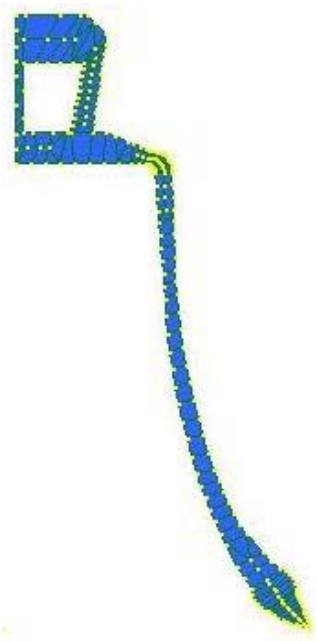


Figure 3-2: Finite element model of half of a typical church bell

The above figure is developed using an estimation of the dimension from figure 2, as the exact dimension of the church bell does not play an important role. This is because it is not a real model to produce a church bell. The figure 3.2 shows only half of a typical church bell, as the profile of a typical church bell is symmetrical. Thus it will be sufficient just to model half of the bell. As the inputs to the software are set to be a full dimension model of a bell, STRAND 7 will take the model as a full model.

Stating the distance of each node and connecting the nodes with plate's element create the finite element, as the figure 3.1 drawing did not specific the exact location of any nodes. By using STRAND 7, the program will automatically divide the length into a specific number of nodes thus creating plates for STRAND 7 to solve for the modal vibration of the bell profile.

Upon completing Figure 3.2, next stage is to convert the drawing into a 3-dimension element model. As a 3-dimension object is drawn on a 2-dimension drawing, it is quite impossible for STRAND 7 to locate the centre of mass because of different coordinates

system (e.g. two dimension only have X and Y axes whereas three dimension have X, Y and Z axes). For STRAND 7, it is easy to change the drawing to a three dimensional model. The figure below shown is the conversion of the model. As mention earlier, modelling only half of the profile is needed because the shape of the bell is symmetrical.

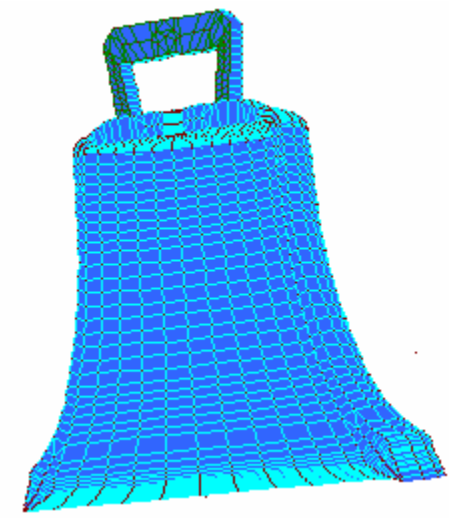


Figure 3-3: 3-D model of the profile of a typical church bell

Since the profile of the bell has a larger bottom part therefore, I would expect the centre of mass to be approximately ¾ from the tip of the bell. From the result of STRAND 7, the centre of mass are approximately 4.7 m from the Y-axis and 2.9 m from the Z-axis after taking the origin at the centre of the sound bow. There is a slight difference in the centre of mass if the bell does not have the top handle as shown in the figure below (Figure 3.4). The centre of mass is lower by 0.1m of the Z-axis, which can be expected, as there is an increase in the mass at the top portion that results in lowering the centre of mass.

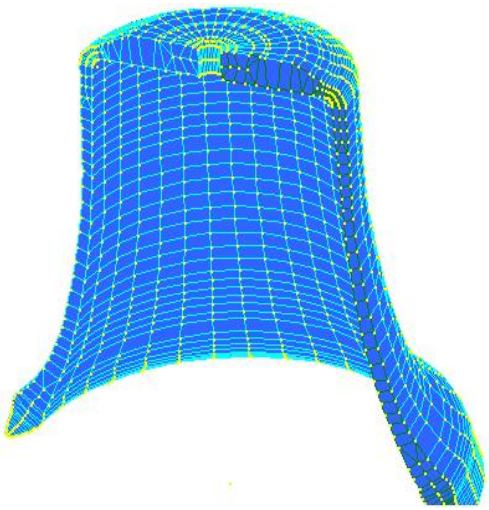


Figure 3-4: 3-D model of the profile of a typical church bell with no top handle

With the constraint given to Figure 3.3, the natural frequency and the mode shapes are calculated and shown in Figure 3.5.

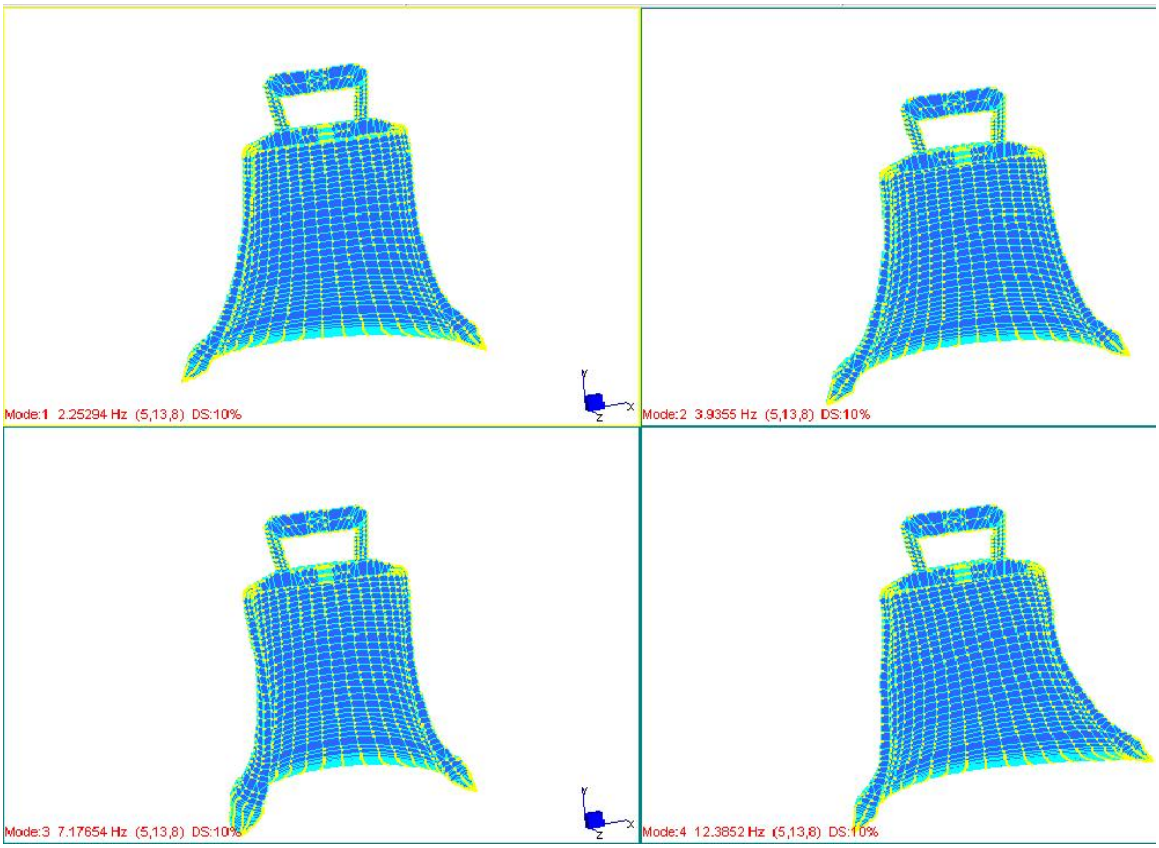


Figure 3-5 Natural frequencies and the mode shape of the bell structure

3.3 The study of pendulum

3.3.1 Simple Pendulum

In order to investigate the motion of the swinging clapper in the bell, first we have to look into the equation of motion of the clapper inside the bell. The clapper will play a crucial role in the dynamics of the whole bell system. The equation of motion for the clapper of the bell is non-linear. It contains a two-degree of freedom, which makes it very complex and difficult to investigate. Therefore, I will go back to the basic to study the fundamental of a simple pendulum, which will provide me with a basic understanding of a simple pendulum. It provides me with the necessary knowledge of how simple pendulums will behave before I can predict the motion of the clapper in the church bell.

In order to make the investigation simpler, the rod attaching the bob for the clapper is replaced with a weightless string. This string that is attaching the bob forms a model of clapper in the bell frame.

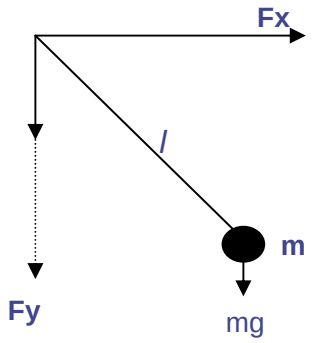


Figure 3-6: Schematic drawing of simple pendulum

Refer to the above figure 3.6; the pendulum composes of an object of mass *m* and a weightless string of a constant length *l* in a constant gravitational field of an acceleration *g*. Even though the motion of the pendulum is a two dimensional, a single generalized coordinates where the angle θ is measured from the negative of Y-axis is needed to

describe the configuration of the pendulum and. Therefore the position of the object is given:

$x(\theta) = l\sin\theta$ and $y(\theta) = -l\cos\theta$

The velocity of both axes is given,

$\dot{x}(\theta,\dot{\theta}) = l\dot{\theta}\cos\theta$ and $\dot{y}(\theta,\dot{\theta}) = l\sin\theta$

For free vibration of an undamped system, the energy in the system is a combination of kinetic energy and its potential energy. The definition of the kinetic energy is the energy stored in the mass by virtue of its velocity and that of the potential energy is the energy stored in a form of strain energy in elastic deformation. The kinetic energy of the pendulum is defined as,

$K = \frac{1}{2}m\left(l\dot{\theta}\right)^2$

When the kinetic energy of the system is found at its maximum, the potential energy of the system should be zero if no other external force applied to the pendulum. Therefore choosing the zero potential energy point when θ=0, and the gravitational potential energy will be;

$U = mgl(1-\cos\theta)$

Using Lagrangian equation L = K - U, the equation is written as;

$L = \left(\frac{m}{2}\right)l^2\dot{\theta}^2 - mgl(1-\cos\theta)$

And the Euler-Lagrange equation for θ is

$\frac{dL}{d\dot{\theta}} = ml^2\dot{\theta}$ and $\frac{dL}{d\theta} = -mgl\sin\theta$

Differentiating $\dot{\theta}$ against time the equation will become

$\frac{d}{dt}\left(\frac{dL}{d\dot{\theta}}\right) = ml^2\ddot{\theta}$

Therefore the equation of motion for the simple pendulum will be,

$$\ddot{\theta} + \frac{g}{l}\sin\theta = 0$$

Apart from using the Energy method to formulate the equation of motion for the pendulum system, another method may be exploited to verify the above equation of motion for the pendulum.

Using the summation of forces on the object mass *m* subject to the pivoted point and the angle θ, the summation of forces in the horizontal direction will be,

$$\sum X = ml^2\ddot{\theta}$$

The forces in the vertical direction will be;

$$\sum Y = -mg\sin\theta$$

The summation of forces in the horizontal direction minus the forces in the vertical direction should be equal to zero,

$$\sum X - \sum Y = 0$$

Therefore, the equation of motion will become,

$$ml^2\ddot{\theta} + mgl\sin\theta = 0$$

$$ml(\ddot{\theta}l + g\sin\theta) = 0$$

$$\therefore \ddot{\theta} + \frac{g}{l}\sin\theta = 0$$

The equation of motion derived using the summation of forces on the mass is similar with the equation derived using the energy method. We could therefore assume that the equation derived is correct.

From this equation, I expected the plot for θ against $\dot{\theta}$ to be circular. When angular position increases, the angular velocity will decreases; the angular displacement increases to a maximum point then it will start to decrease. This is when the pendulum reaches the point where the kinetic energy will be zero with maximum potential energy. As the pendulum will start to move to the other direction, the kinetic energy will begin to rise and the potential energy will start to drop, thus causing the plot to form a semi circular shape. Hence, when the pendulum tried to return to its original position, the plot of θ against $\dot{\theta}$ will become circular.

As the equation of motion for the pendulum has derived, the other important fields to look into are the frequency and the period of the simple pendulum. Both the frequency and the period are also the fundamentals of the system.

For a simple pendulum the formula to determine the frequency is,

$$f = \sqrt{\frac{g}{l}}$$

where *g* is the gravitational pull (9.81m/s) and *l* is the length of the pendulum.

From the equation, we can predict that as the length of the pendulum increases, the frequency of the pendulum will become smaller and vice versa. The length of the pendulum will determine whether the pendulum will have a high or low frequency.

For a simple pendulum with small oscillation, the period τ is defined as,

$$\tau = 2\pi\sqrt{\frac{l}{g}}$$

This formula only applies to small angle as $\sin\theta$ is approximately equal to θ, but I am not looking at small angle so this equation may not be very useful. In order to use this

equation, a correction factor must be added to the equation. Therefore, the formula for the period τ will be defined as,

$$\tau = \frac{2k}{\pi}\left(2\pi\sqrt{\frac{l}{g}}\right)$$

Where k is the correction factor, it will vary as the pendulum subject to different oscillation. For those angles that is less than 10 degrees considered small angle, which the correction factor could be ignored.

Angle (in degree)	0	10	20	30	60	90	120	150	180
k (correction factor)	1.571	1.574	1.583	1.598	1.686	1.854	2.157	2.768	∞

Table 1: Correction factor for large oscillation (source: Kidd and Fogg 2002)

From the table 1, one would notice that as the angle of oscillation increases, the value for k increases until it reaches around 150 degree. Any angle of oscillation greater than 150 degree, the pendulum would be unstable and it would be quite difficult to estimate the period.

From the equations of the frequency and the period for a simple pendulum, we could predict that for the pendulum, the larger the angle of oscillation, the larger the period will be. The frequency will not change if the length of the pendulum is sustained. As the frequency is only dependent on the length of the pendulum, increasing the angle of oscillation will not increase the frequency.

3.3.2 Simple Double Pendulum

After gathering the basic knowledge on the simple pendulum, a simple double pendulum will be studied. It will have similar characteristics as the clapper of the church bell. Now the pivoted point is lower and an object mass will be placed above this point to stimulate the model of a bell. The schematic diagram is show in figure 6.

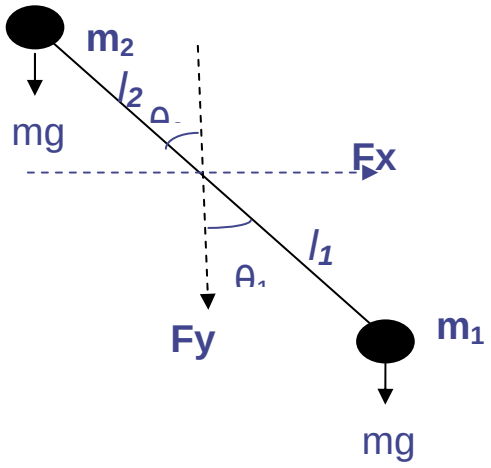


Figure 3-7: Schematic diagram of a simple double pendulum

For this model, the angle of oscillation for the two masses is the same as they rotate at the same point. Therefore θ_1 and θ_2 should be equal regardless of the distance between the masses.

Since this is a similar system as the simple pendulum model, it will be easier to derive the equation of motion using the forces summation method rather than energy method. The equation of motion for the simple double pendulum will be,

$$m_1 l_1 \ddot{\theta} + m_2 g \sin\theta = m_1 g \sin\theta + m_2 l_2 \ddot{\theta}$$

$$g \sin\theta \left(m_2 - m_1\right) = \ddot{\theta}\left(m_2 l_2 - m_1 l_1\right)$$

$$\therefore \ddot{\theta} = \frac{g \sin\theta \left(m_2 - m_1\right)}{m_2 l_2 - m_1 l_1}$$

From the equation of motion, I would expect that as either of l_1 or l_2 increases while the masses remain the same, the angular acceleration would decrease. A sudden increase in the masses while maintaining the lengths, the angular acceleration would increase. The

angular acceleration is dependent on both the lengths and the masses. A change in either one will affect the angular acceleration.

As the angle for θ_1 and θ_2 is the same, I can assumed that the plot for θ against $\dot{\theta}$ will be similar to the one as described for a simple pendulum. Only when there is a change in the physical properties of the system will cause a change in the acceleration but the velocity and the displacement will be similar to the one discussed in section 3.3.1.

The frequency of the simple pendulum is only affected by the length between the pivoted point and the object mass. The angle of oscillation for the simple pendulum does not have any effect on the frequency. I can assume that the equation of frequency for the simple double pendulum will be similar as the simple pendulum, which is,

$$f = \sqrt{\frac{g}{l}}$$

However, on the simple pendulum system there is only one length involve. Whereas for this system there are two lengths instead of one, hence the equation of frequency will become,

$$f = \sqrt{\frac{g\left(m_2 - m_1\right)}{m_2 l_2 - m_1 l_1}}$$

The mass in the system will also play an important part in determining the frequency of this system. For the simple double pendulum system, it will have a larger field of physical parameter to depend on while determining the natural frequency of the system.

The equation for period of the simple double pendulum will be 2π divide the natural frequency, that is,

$$\tau = 2\pi\sqrt{\frac{m_2 l_2 - m_1 l_1}{g\left(m_2 - m_1\right)}}$$

Similar to the simple pendulum equation, to reduce the error of the system, a correction factor will placed into the equation. Hence changing the equation to,

$$\tau = \frac{2k}{\pi}\left(2\pi\sqrt{\frac{m_2 l_2 - m_1 l_1}{g\left(m_2 - m_1\right)}}\right)$$

Using the same table 1, it will enable one to calculate the period of the simple double pendulum of the system. From the equation, we can also predict that the higher the value of the correction factor *k* will cause an increase the period of the system.

3.3.3 Double pendulum

Upon the completion of the investigation of the simple double pendulum, I should be equipped with sufficient knowledge to determine the dynamic of church bell. Before I move on to the study of the church bell, I would bring the attention to look into the double pendulum first. From my understanding of the different type of bell profile, none of them will use the double pendulum arrangement for the bell system. This is due to the double pendulum system is very unstable and the small errors will tend to grow exponentially creating a more chaotic motion (Fletcher, McGee and Tarnopolsky 2002: 437-1444).

Even though the motion of the double pendulum and that of the church bell are different, there are similar mechanical properties (e.g., both the system will have two different pivoted points and there are two angles of oscillation for the systems). Therefore, the investigation of the equation of motion of the double pendulum helps me to understand and to develop the equation of motion for the mechanical model of the church bell.

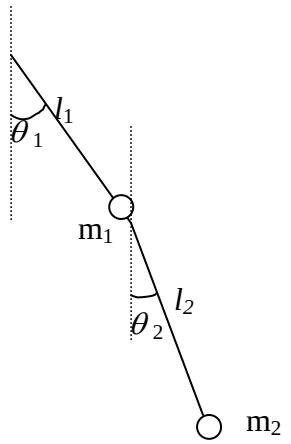


Figure 3-8: Setup for double pendulum

Figure 3.8 is a diagram of the double pendulum that consists of two sections. Firstly, for m_1, it is pivot on a fixed point. From the fixed point, the distance between m_1 and the pivot point is l_1 it is allowed to vary its orientation by θ_1 from the vertical. Then, a second mass m_2 is suspended a further distance l_2 from the first mass. It has an angle of freedom being θ_2 from the vertical at any instant.

Using Lagrange's equation, the kinetic energy of the system defines as,

$$T = \frac{1}{2} m_1 l_1 w_1^2 + \frac{1}{2} m_2 \left(l_1 w_1 + l_2 w_2 \right)^2$$

In addition, the equation for potential energy for the system is,

$$U = m_1 g l_1 (1 - \cos\theta_1) + m_2 g l_2 (1 - \cos\theta_1 + 1 - \cos\theta_2)$$

$$\frac{d}{dt}\left(\frac{dT}{dw_1} \right) = 2 m_1 l_1^{\,2} \dot{w}_1 + m_2 l_2^2 \dot{w}_2 \,,\ \frac{d}{dt}\left(\frac{dT}{dw_2} \right) = m_2 l_2^{\,2} \dot{w}_2 + m_1 l_1^{\,2} \dot{w}_1$$

$$\frac{dU}{d\theta_1} = g\left(m_1 l_1 \sin\theta_1 + m_2 l_2 \sin\theta_2 \right),\ \frac{dU}{d\theta_2} = m_2 l_2 \sin\theta_2$$

If *w* is the angular velocity $\dot{\theta}$, using

$$\frac{d}{dt}\left(\frac{dT}{dw}\right)+\frac{dU}{d\theta}=0$$

The equation of motion for double pendulum will be,

$$2m_1l_1^2\ddot{\theta}_1+m_2l_2^2\ddot{\theta}_2+g\left(m_1l_1\sin\theta_1+m_2l_2\sin\theta_2\right)=0 \text{ and } m_2l_2{}^2\ddot{\theta}_2+m_1l_1{}^2\ddot{\theta}_1+m_2l_2\sin\theta_2=0$$

Rearranging the whole equation, we get,

$$\ddot{\theta}_1=\frac{-g(m_1l_1\sin\theta_1+m_2l_2\sin\theta_2)-ml_2^2\ddot{\theta}_2}{2m_1l_1^2} \text{ and } \ddot{\theta}_2=\frac{-m_1l_1{}^2\ddot{\theta}_1-m_2l_2\sin\theta_2}{m_2l_2{}^2}$$

The objective of deriving the equation of motion for the double pendulum is to assist me in deriving the equation of motion for the clapper. After deriving the equation of motion for the double pendulum, it will be easy to find the angular displacement, which is the determinant of the equation of motion. As the process to find the determinant of the equation is quite lengthy, I will not continue to solve for θ and eventually solve for the frequency and the period for the double pendulum.

3.3.4 Analytical model of the bell

On the completion of all the different type of pendulums, I have acquired sufficient knowledge to derive the equation of motion for the bell. The equation of motion for the church bell is non-linear. There are two cases which I need to consider, one of them is when the bell and the clapper moves separately, and the other is when they are in contact and have the same velocity. Figure 3.9 is a schematic diagram of the mechanical arrangement of a church bell.

When the clapper and the bell move separately, the church bells have two degrees of freedom, which could easily represent by the rotation of the bell itself, θ_1 and the clapper, θ_2. An expression for the kinetic energy and potential energy of the system can be determined and the equation of motion may be derived using Lagrange's equation. The Lagrange's equation used for the bell model is expected to be quite similar to the double pendulum.

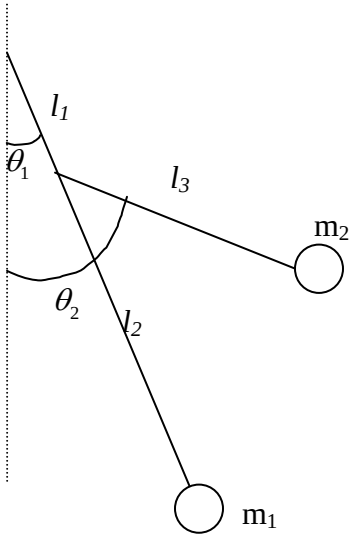


Figure 3-9: Schematic diagram of the mechanical model of church bell

Using the same method for the double pendulum, the kinetic energy for the model of the church bell is,

$$T = \frac{1}{2} m_1 \left[\left(l_1 + l_2 \right) \dot{\theta}_1 \right]^2 + \frac{1}{2} m_2 \left(l_3 \dot{\theta}_2 + l_1 \dot{\theta}_1 \right)^2$$

In addition, the potential energy will be,

$$V = m_1 g \left(l_1 + l_2 \right) \left(1 - \cos\theta_1 \right) + m_2 g (l_1 + l_3) \left[\left(1 - \cos\theta_1 \right) + \left(1 - \cos\theta_2 \right) \right]$$

$$\frac{d}{dt}\left(\frac{dT}{d\dot{\theta}}\right) = m_1\left(l_1 + l_2\right)^2 \ddot{\theta}_1 + m_2 l_3 \ddot{\theta}_2 + m_2 l_1^2 \ddot{\theta}_1 \text{ and } \frac{d}{dt}\left(\frac{dT}{d\dot{\theta}}\right) = m_2 {l_3}^2 \ddot{\theta}_2 + m_2 l_1 \ddot{\theta}_1$$

$$\frac{dV}{d\theta_1} = m_1 g\left(l_1 + l_2\right)\sin\theta_1 + m_2 g\left(l_1 + l_3\right)\sin\theta_1 \text{ and } \frac{dV}{d\theta_2} = m_2 g\left(l_1 + l_3\right)\sin\theta_2$$

Using,

$$\frac{d}{dt}\left(\frac{dT}{d\dot{\theta}}\right) + \frac{dU}{d\theta} = 0$$

The equation of motion for church bell will be,

$$m_1\left(l_1 + l_2\right)^2 \ddot{\theta}_1 + m_2 l_3 \ddot{\theta}_2 + m_2 l_1^2 \ddot{\theta}_1 + m_1 g\left(l_1 + l_2\right)\sin\theta_1 + m_2 g\left(l_1 + l_3\right)\sin\theta_1 = 0$$

$$m_2 {l_3}^2 \ddot{\theta}_2 + m_2 l_1 \ddot{\theta}_1 + m_2 g\left(l_1 + l_3\right)\sin\theta_2 = 0$$

By assuming, the angle of swing to be large will make the whole equation of motion more complex. In order to reduce the complexity of the equation, θ is assumed to be small and $\sin\theta \approx \theta$

Let $\theta_1 = A\sin wt$ and $\theta_2 = B\sin wt$

$\ddot{\theta}_1 = -Aw^2 \sin wt$ and $\ddot{\theta}_2 = -Bw^2 \sin wt$

The equation will become,

$$-Aw^2 m_1\left(l_1 + l_2\right)^2 - Bw^2 m_2 l_3 - Aw^2 m_2 {l_1}^2 + Am_1 g\left(l_1 + l_2\right) + Am_2 g\left(l_1 + l_3\right) = 0$$

$$-Bw^2 m_2 {l_3}^2 - Aw^2 m_2 l_1 + Bm_2 g\left(l_1 + l_3\right) = 0$$

In matrix form,

$$\begin{pmatrix} m_2 g\left(l_1+l_3\right)+m_1 g\left(l_1+l_2\right)-w^2 m_2 l_1{}^2-w^2 m_1\left(l_1+l_2\right)^2 & -w^2 m_2 l_3 \\ -w^2 m_2 l_1 & m_2 g\left(l_1+l_3\right)-w^2 m_2 l_3{}^2 \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix}=0$$

As the determinant of the equation will be very lengthy and complex, MATLAB is used. The determinant will be,

$$p^2+pqg-w^2 sp-w^2 q(l_1+l_2)-prw^2 l_3-rqgw^2++w^4 rsl_1 l_3+w^4 qr(l_1+l_2)l_3-w^4 rs$$

For p = gm_2 (l_1 + l_3), q = m_1 (l_1 + l_2), r = $m_2 l_3$, s = $m_2 l_1$

Rearrange,

$$w^4\left(rsl_1 l_3+qr(l_1+l_2)l_3-rs\right)-w^2\left(sp+q(l_1+l_2)+prl_3+rqgl_3\right)+p^2+pqg$$

Therefore,

$$w=\Big(sp+q(l_1+l_2)+prl_3+qrgl_3\pm\Big(2q^2 l_1 l_2+q^2 l_1{}^2+q^2 l_2{}^2+s^2 p^2+2l_1 c+2l_2 c+2l_3 d+$$

$$b^2 l_3^2+2agl_3+2bl_3 l_1+2q^2 rgl_3 l_1+2bl_3 l_2+2q^2 rgl_3 l_2+2brl_3^2 g+p^2 r^2 l_3^2-4dl_3 l_1-4agl_3 l_1-$$

$$4bpl_3 l_1-4bqgl_3 l_1-4bpl_3 l_2-4bqgl_3 l_2+4d+4ag\Big)^{\frac{1}{2}}\Big/\left(2rsl_3 l_1+2qr(l_1+l_2)l_3-2rs\right)\Big)^{\frac{1}{2}}$$

Where, a = pqrs, b = pqr, c = spq, d = rsp^2

Upon determining the value for *w,* it will be easy to determine the value for the frequency and the period.

The frequency of the church bell will be,

$$f_n = W_n$$

Analytical methods are used to prove that the equations derived above are correct. The geometry of the profile of the bell will incorporate into numerical method to find out the behaviour of the bell when it is subject to a single strike on the bell frame. It will be described in the next chapter to verify all the equations that derived in this chapter are true.

4 Numerical analysis for pendulum

4.1 Introduction

After deriving all the equations, it is yet to verify that the equation derived is acceptable. There is no doubt that the equation of motion for the pendulum system derived using the energy method is being compared with the one using the summation of forces method. Both the methods maybe correct but human errors such as the substitution of the equation and the derivation of the initial equation is incorrect. If the initial equation is already incorrect, this may lead to the whole equation to be incorrect.

In order to avoid unnecessary human error, the equation is being ‘tested’ using MATLAB. Another advantage is to compare the expected result using analytical analysis and numerical analysis and keep a lookout for any discrepancy. It is not a very good approach to use manual calculation to solve for the equation of motion, which derived earlier. Since numerical method is used for the calculation in the equations, it would be easier to enhance the program developed just for the calculations to a more versatile MATLAB program for future other similar calculation.

Section 4.2 gives an explanation on how the MATLAB program is developed. In section 4.3, the program will test using the equation derived in the previous chapter to ensure that both manual calculation and numerical solver is true. This is to provide a basic understanding of the fundamental that is affecting the swinging pendulum and how does the motion of a church bell looks like.

4.2 Development of MATLAB program

There are two approaches to develop this program. Either writing a script using the "m-file editor", which is part of MATLAB program function, or construct a GUI (Graphical User Interface) model. For the script writing, it will be useful if the user is quite familiar with MATLAB, if not the user would not be able to use. It would be a problem for anyone who tried to use the program, and is not very familiar with MATLAB programming. It may be more complex to develop a GUI program, but it is very user-friendly as any anyone would be able to use regardless of whether the user is familiar with MATLAB programming.

The main advantage of GUI program is that for users would not have to compile the program repeatedly to obtain check for result and to view the plots. It would be more convenient for users who wish to have multiple inputs or multiple different perimeters to test. By using a GUI model, users can enter the perimeters by clicking onto the respective location and the results will be plotted on the screen.

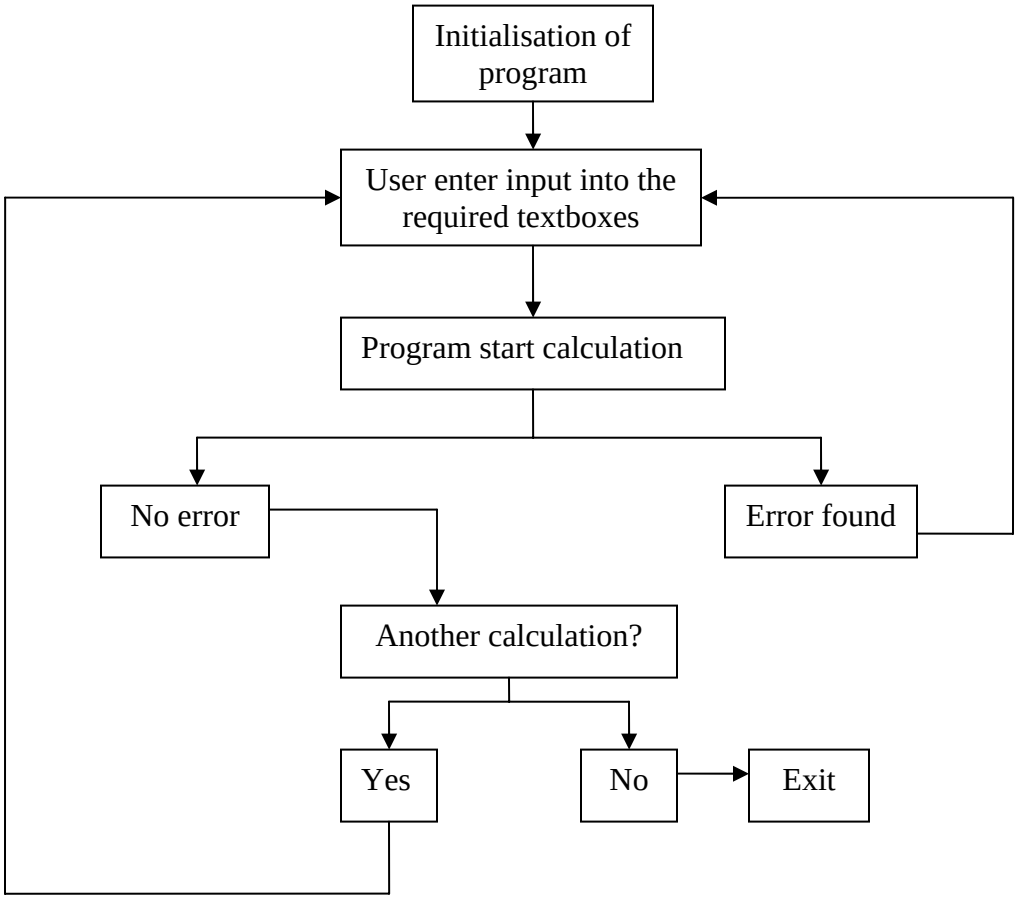
Before developing the GUI model, the parameters required in the GUI and types of plots need to be shown to the user must be known. All the minor details have to decide before building any models. Blueprints for the construction of the GUI model needs to be consider (e.g. the position of the buttons and the input parameters etc).

The program is based on the specification stated below:

- The program should be object oriented
- The program should be written in MATLAB
- GUI must be implemented
- Proper error messages should be generated
- All necessary input textboxes must be visible
- There should be a way for the user to exit the program

The working function of the program is when the program is loaded; the user will have to enter all the inputs in the space provided. Once all the inputs have been entered, click on the ‘Enter’ button on the model, the program will calculate the frequency and the period based on the entered parameters.

If the angle of oscillations exceeded 150 degrees, an error message will prompt the user that the period will have gone to infinite. However, the rest of the other outputs are show on the screen. The other error message of the whole program is when the user entered an alphabet into any of the perimeter the program will prompt out an error saying that, that field is only for number. To have another calculation the user will needed to change the number in the respective textboxes. The program will only exit, when the user click on the ‘Close’ button or click on the upper right-hand corner ‘cross’ icon. Below is a flow chart showing how the program works



After stating all the specification on the program, the basic design of the program is shown in figure 4.1.

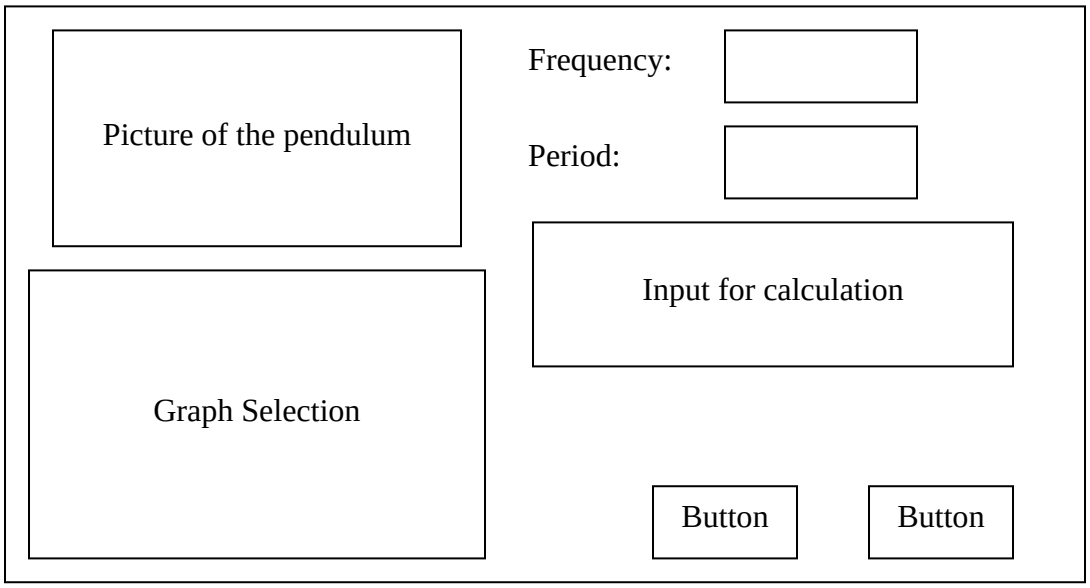


Figure 4-1: Layout of the GUI model

To avoid confusion of the input, an image of the pendulum will be placed in the GUI model; this is to enable the user to identify which physical perimeter that I am referring. The input of the physical perimeters is placed at the bottom of the model while the two characteristics of the equation (natural frequency and the period) are located at the top right hand of the model. By doing the plot, I could check whether the equation of motion, which I have derived, is correct as compare with the actual movement of the pendulum. Instead of using one plot, I have developed a selection of graphs, which allow the user to have a better understanding of the behaviour of the pendulum.

The MATLAB GUI model that was developed is compare with the analytical methods. Before any comparison, it is necessary for the program developed is running correctly and error free. The program is set on a series of tests to ensure its reliability.

4.3 Numerical vs. analytical

4.3.1 GUI program for simple pendulum

Figure 4.2 shows an illustration of the GUI model. By entering the angle of swing and the desired length, the program will calculate the frequency and the period based on the physical data. A graph will be plotted.

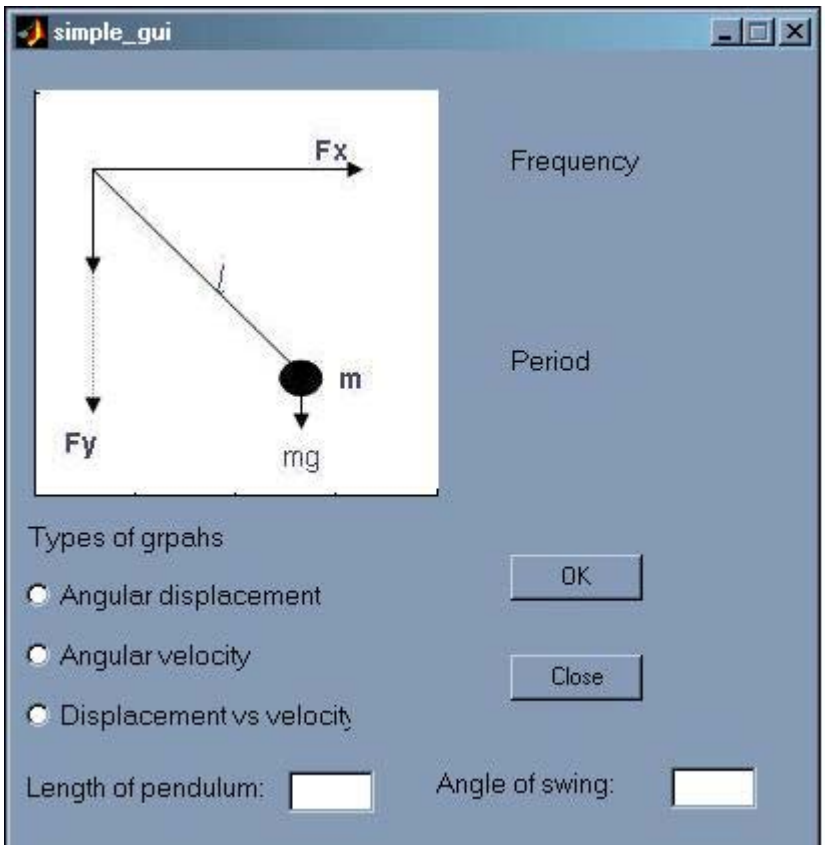


Figure 4-2: GUI model for simple pendulum

From figure 4.3 which show the motion of the pendulum for a time-span of 5 sec, we could justify that the equation derived in the previous chapter is correct. From the plot, we would notice that the angle will reach zero and increased in the negative direction. The MATLAB will take that the starting angle to be positive and the pendulum will start to travel down. It will swing from left to right and back to its initial starting position.

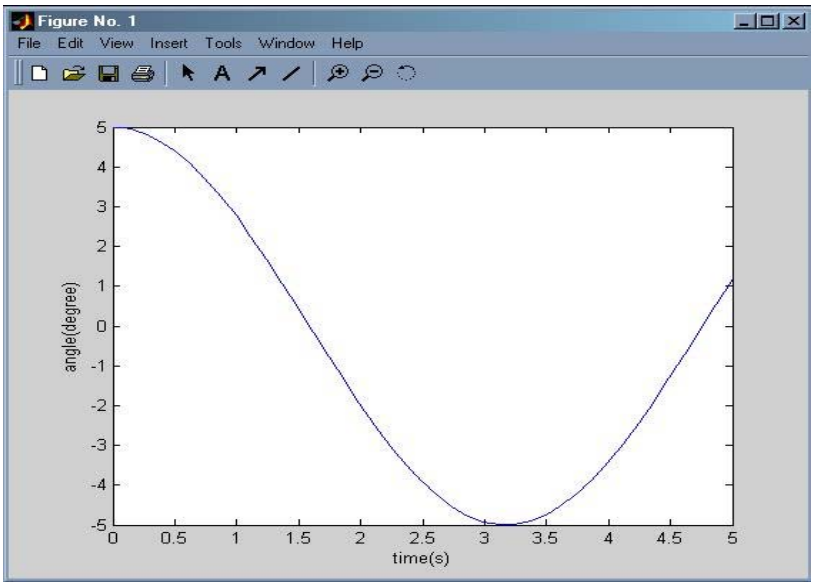


Figure 4-3: Simple pendulum with small angle

By my intuition, as the angle of swing increases, the pendulum will take a longer time to complete a single swing. However, as the frequency remains the same, it shows that the equation, which obtained from the previous chapter, is true. From the result, there is a slight increase in the period, as it will depend on the constant k.

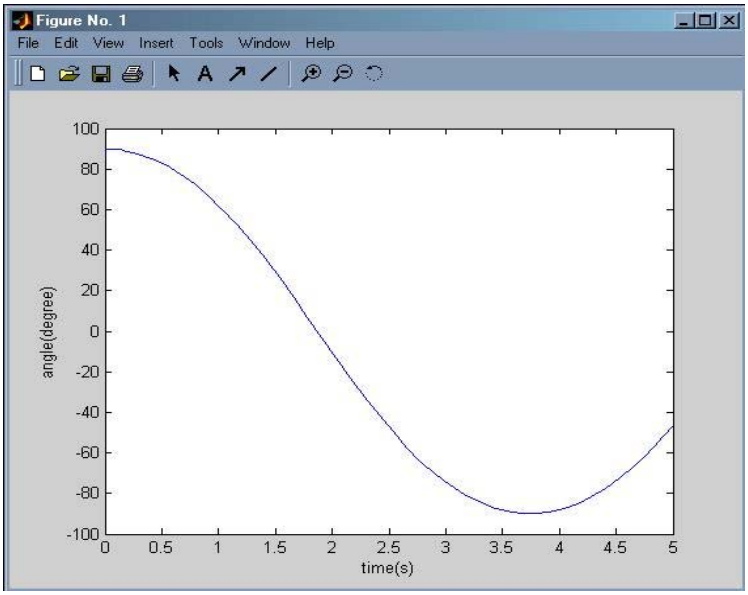


Figure 4-4: Simple pendulum with large angle

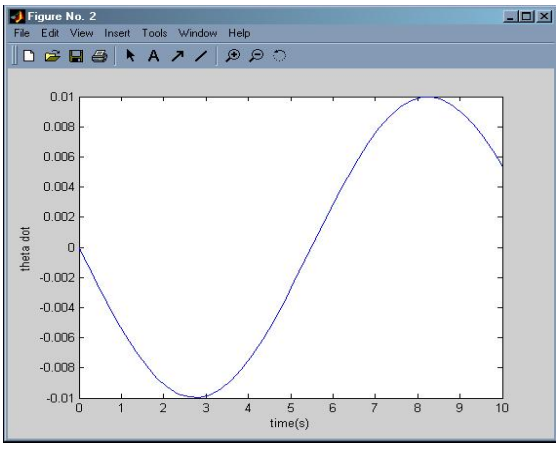


Figure 4-5: Plot for angular velocity for small angle

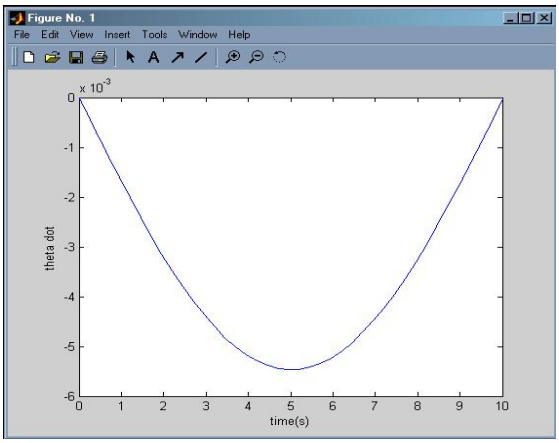


Figure 4-6: Plot for angular velocity increased length

Comparing both the figure 4.5 and the figure 4.6, it is shown that when the pendulum remains at small angle and if its length is increased, its angular velocity will tend to increase significantly.

From all the plots, I can conclude that the longer the length the pendulum will cause the angular velocity to increase. The pendulum will need to take a longer time to return to its initial position. The numerical result fits well with the analytical result, which indicates that the equations derived in the pervious chapter are correct. For this is a simple model, it is quite easy to derive the equation, as there are only two perimeters. As the number of perimeters increases and it would become tougher and the equation will become more complex

4.3.2 GUI for simple double pendulum

Figure 4.7 shows the GUI model for the simple double pendulum where more parameters are added. Similar to the previous GUI model, upon entering the desired lengths and the angle of swing, the program will be able to calculate the period, the frequency and a plot of theta vs. time is being plotted.

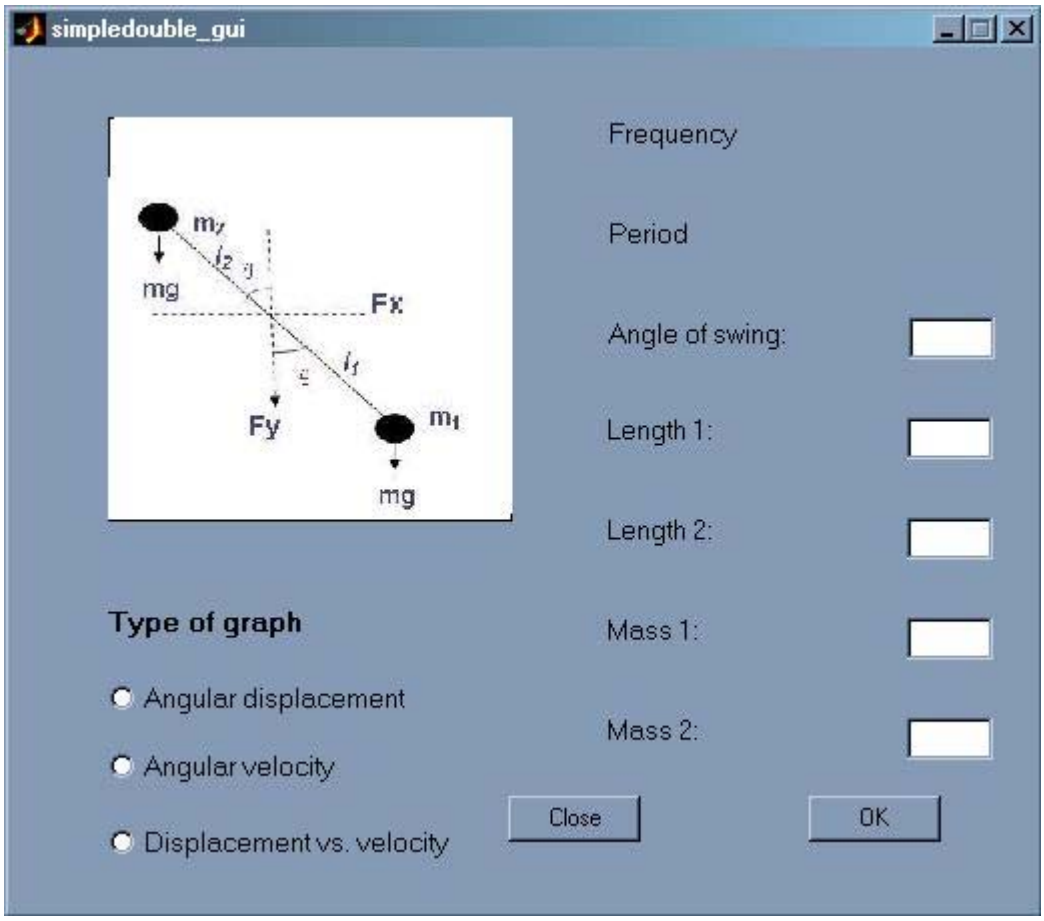


Figure 4-7: GUI model for simple double pendulum

Since the equations for both the equation of motions are different, it is expected that the plot for angular velocity will be different to the one for simple pendulum. For the pivoted point is lower and with an additional mass m_2, the pendulum will react differently as the top mass will try to 'push' the pendulum to a higher angle as it oscillate.

By my intuition, the value for l_1 and the value for m_1 will be much greater than m_2 and l_2 regardless of the value for the angle of oscillation. As for m_2, it is a stimulation of the mass above the pivoted point on a simple hand bell.

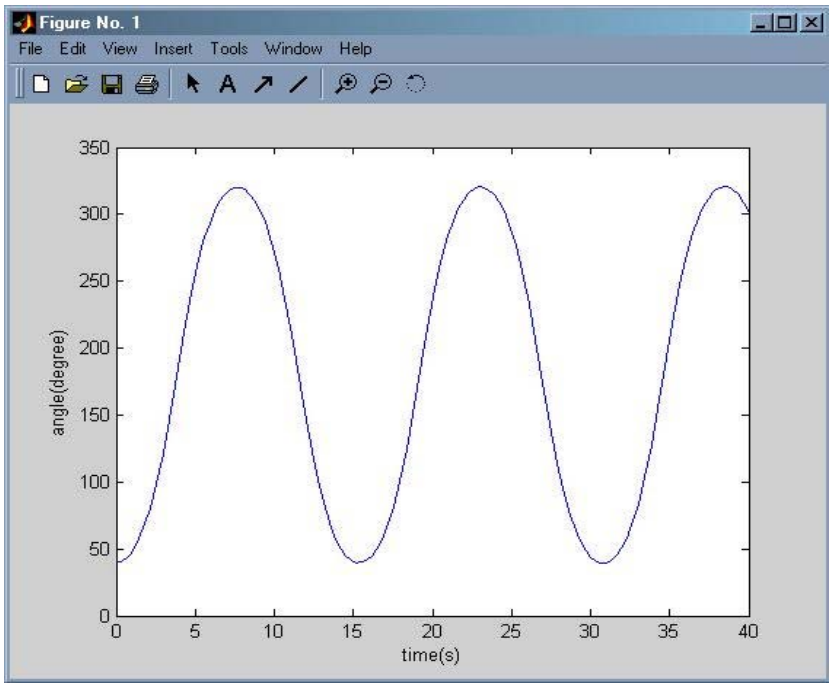


Figure 4-8: Plot on simple double pendulum

Figure 4.8 shows a diagram on stimulation for the motion of the pendulum in a bell, which has no clapper attached in it. I have assumed that the starting angle of oscillation to be 40 degrees and the pendulum is subject to a time span of 40 sec. From the plot, we would notice that the angular displacement would travel upwards, instead of going downwards due to gravitational pull m_1. As the angle of oscillation gets larger, m_2 will be pulled down causing the system to move counter-clockwise thus increasing the angular displacement. For an initial angle of 40 degrees, the system would oscillate to an angle approximately 320 degrees. This phenomenon only happens to free vibration without any constraint.

As the bell frame restricts the pendulum from moving upwards, m_1 would travel downwards to hit the surface of the bell creates an impact and return to its initial starting position. The plot for angular displacement shown on figure 4.9 will be similar to figure 14.

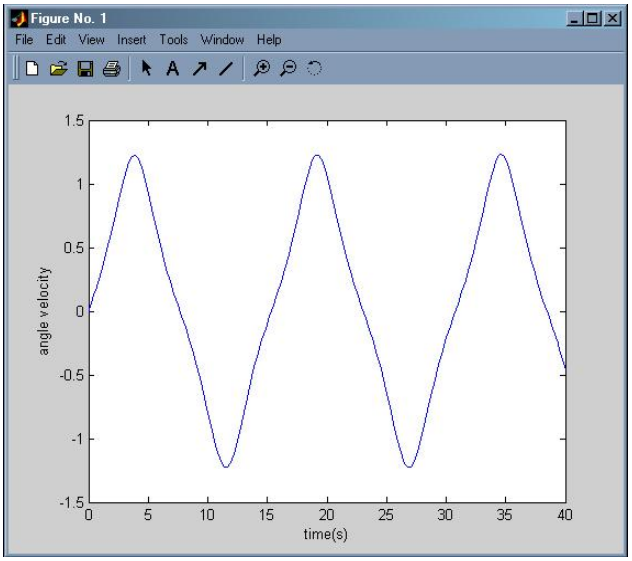


Figure 4.9

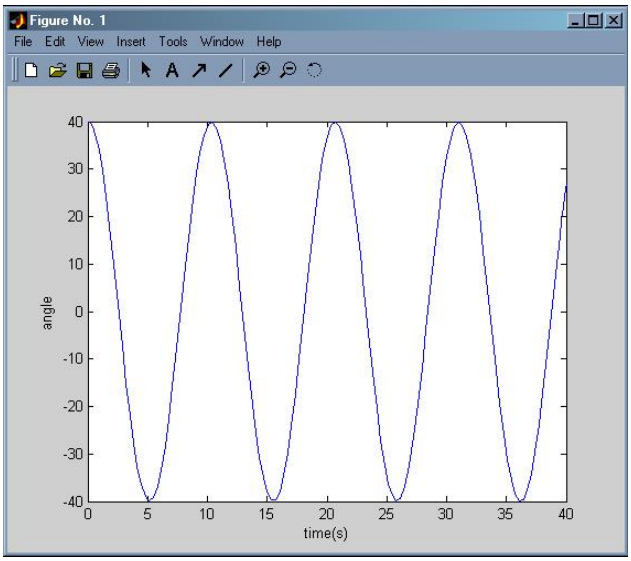


Figure 4.10

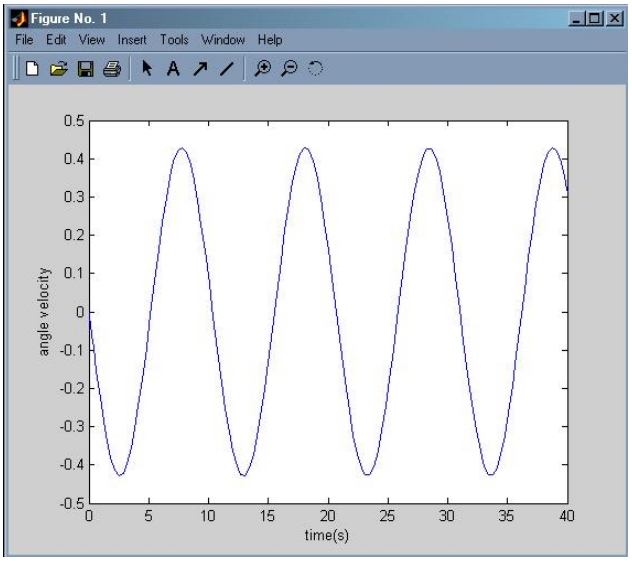


Figure 4.11

Figure 4.9: Plot on angular velocity on free vibration for simple double pendulum.

Figure 4.10: Plot on angular displacement on force vibration for simple double pendulum.

Figure 4.11: Plot on angular velocity on force vibration for simple double pendulum.

4.3.3 GUI model on clapper and bell

Figure 4.12 shows the GUI model of the clapper. The GUI model will be quite similar to all previous models. The equation of motion for the church bell is non-linear, and it has a two-degree of freedoms. Therefore for this model instead of presenting the calculation for the period, the model will show the two different natural frequencies for the entered physical parameters. Since there are two different angles in the system, the plot illustrates the motion of swing for both the clapper and the bell.

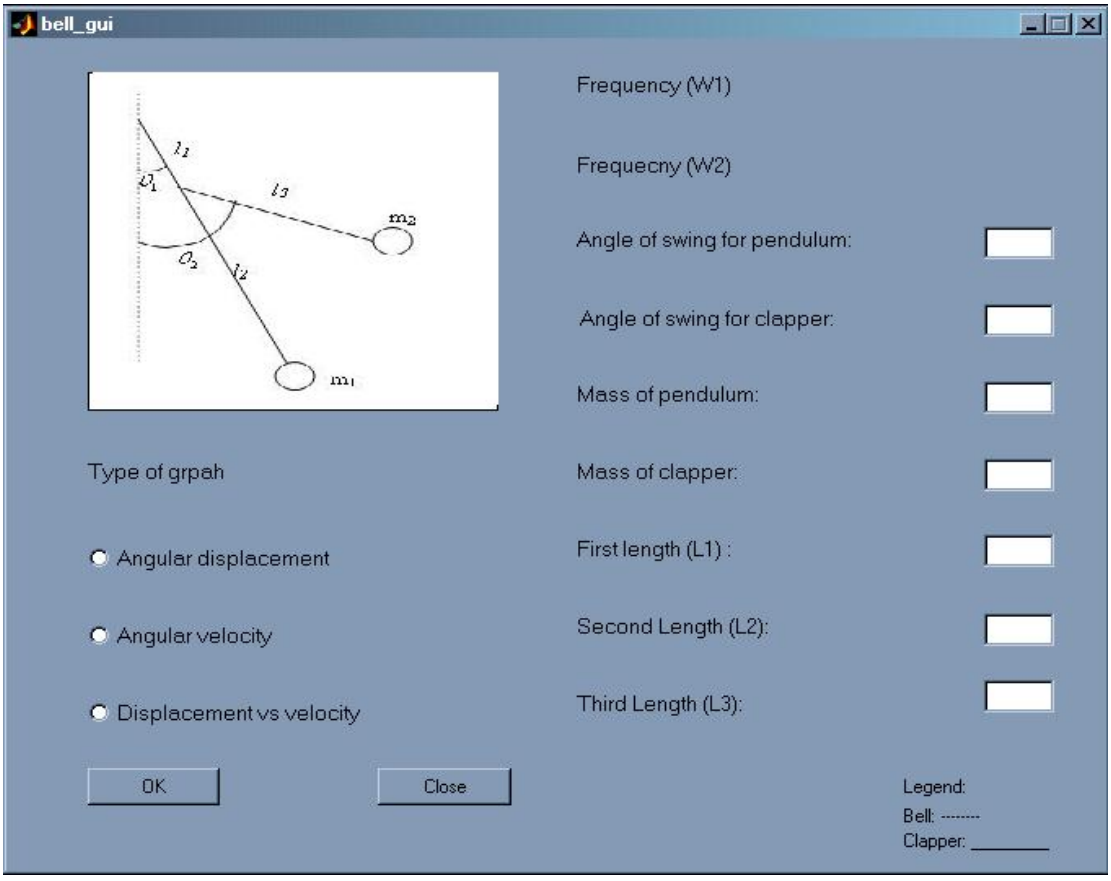


Figure 4.12: GUI model for clapper

Using figure 3.1, the initial condition for θ_1 is approximately 30 degrees when the church bell is in 'set' position. Since the clapper is resting on the side of the bell frame, we could assume that the initial conditions for both the clapper and the bell are the same. However for the real model, it is impossible for both the clapper and the pendulum to have the

same initial condition thus the angle for the pendulum will be offset by around 2 degrees to stimulate the real life model. As some of the physical data for the bell is unavailable, therefore an estimation of the physical parameters will be allocated for the equation of motion. For modelling purposes, we assume that the clapper is working in ideal condition.

Angle of swing for pendulum	32 degree
Angle of swing for clapper	30 degree
Mass of pendulum	31.9 kg
Mass of clapper	10.25kg
Length between the two pivot point	5 m
Length from pivot point to the pendulum	12 m
Length from the pivot point to clapper	20 m

Table 2: Configuration of bell (source: Blakeborough 2001: 69-92)

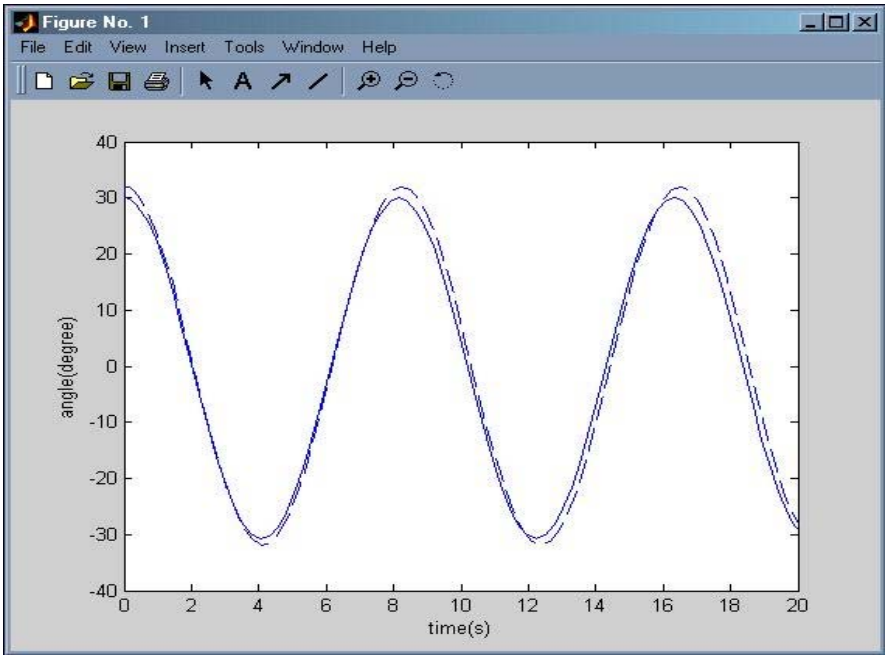


Figure 4.13: GUI using real bell model configuration

In reality, sometimes the clapper will tend to rebound and perform a double hit or swing back striking the back face of the bell. This phenomenon is difficult to stimulate using any numerical solver; therefore, we assume that the clapper is swinging without any abnormality. In figure 4.13, we see that the clapper will move roughly together with the bell. The point of separation for the clapper from the sound bow could be calculated by the impulse response to the bell. This will also give some insight into the properties of the motion of the bell. The natural frequencies for the bell configuration are 0.51Hz and 0.73Hz.

An impact will occur when the clapper hits the sound bow and generating a sound. There are three different kind of impact, which we need to pay attention to,

- when the clapper is in contact with the sound bow
- begin to separate from the sound bow, and
- the time before any contact with sound bow.

The time when the bell and the clapper are in contact and the time of separation of the clapper from the bell can be determined by considering the accelerations that the clapper and the bell would be moving separately. The point of separation was determined by linear interpolation of the acceleration over the time, if the accelerations indicated that the clapper would be accelerating away from the bell.

The direction of the clapper will move in the opposite direction due to impact. In order to put this configuration into the GUI model, as shown in Figure 4.12, we assume an impact would have occurred at around 30 or -30 degrees. When the angle of oscillation for the clapper reaches approximately 30 degrees, the value for the velocity will change either from a positive value to a negative value or vice versa.

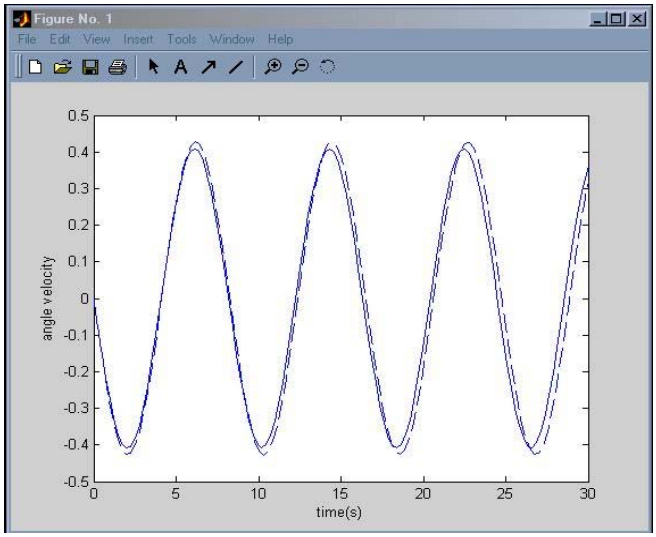


Figure 4.14: Angular velocity without impact

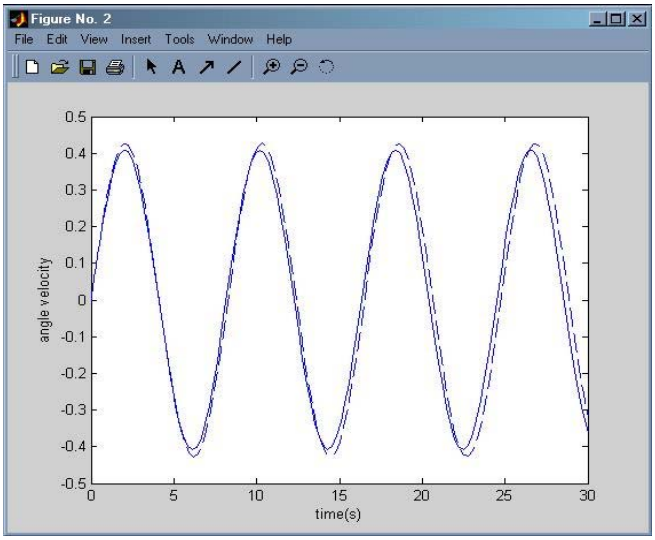


Figure 4.15: Angular velocity with impact configuration

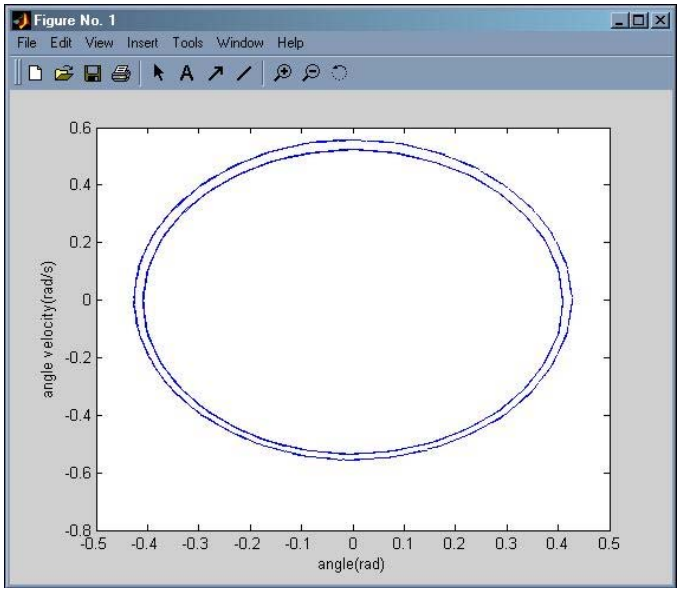


Figure 4.16: Displacement vs angular velocity

5 Concluding Remarks

5.1 Introduction

In this chapter, we want to summarize the outcomes of this thesis and to discuss some ways for further studies. The next section provides a summary of the preceding chapter. After that, we will discuss the possible ways for further studies regarding the dynamics of church bell (section 5.3). In the subsection 5.3.1, we discuss further regarding the studies of the application of the impact theorem on the clapper. The focus of subsection 5.3.2 is on the MATLAB model of the previous chapter and the enhancement of the MATLAB GUI model will be discussed. In the subsection 5.3.3, the modal analysis of the profile of the church bell will be discussed.

5.2 Summary

In the first part of the thesis, it gave an overview of the study to a church bell and the reasons for taking up this thesis. It also highlighted the key objective of the whole thesis.

The second part of the thesis gave a brief history of church bell; why and how the first English church bell was introduced to England. The profile of the bell was made of cast iron but there were some difficulties faced by the bell manufacturer. The differences between the hammered church bell and the clapper-type church bells were looked into, and hence the reasons why the clapper-type church bell was preferred. The different parts and the function for each component were introduced. There was also a short account on when the pulley is pulled from the church bell how does the bell really work. The main objective was to provide a brief understanding on the profile and the modification done to the existing church bell.

Then, the centroid of the clapper-type church bell found by using finite analysis software (STRAND 7). In order to have a better understanding on the dynamic of the bell, the

different type of pendulums relating to the clapper-type church bell was studied. The equation of motion, the period and the natural frequency were derived. Upon deriving the equation of motion, a GUI model was developed. The GUI model will provide a diagram for the user to identify which physical parameter is to be entered. The program will calculate the natural frequency and the period, and a plot for the motion of the pendulum is plotted. There will be some error messages in the program to prevent any miscalculation and typo error from the user. The objective is to compare the result using analytical method and the numerical method.

There are numerous possibilities to extent the analyses of this paper. This was discussed in the previous chapter.

5.3 Further studies

5.3.1 Impact theorem based on the dynamic of bell.

In this paper, the dynamics of the clapper in the church bell is being investigated. However in order to create a sound from the sound bow of the church bell, an impact must occur which required further investigation.

An impact that will occur when the clapper hit both side of the sound bow and at the point of contact will create a sound. Different point of contact will produce different musical notes and the amount of impact will determine the tone created by the bell. Upon understanding the impact theorem, it will incorporate into the equation of motion of the clapper derived in this paper to have a better knowledge of the dynamic of the church bell.

5.3.2 MATLAB enhancement

The outlook of the GUI model can be further modified to become more user-friendly. A menu window, which can access to different program for different model is a good start.

On the opening of the menu, the user will be able to choose the type of calculation that the user required. Instead of entering the data repeatedly, the programmer may develop a slider toolbar which enable the user to enter just once and drag along it to see the different effect on the frequency and the period of the system if a certain parameter decreases or increase.

5.3.3 Modal analysis of the profile

The clarity of the sound produced by the bell is dependent on the profile of the bell. On the paper, there was not much study done based on the modal analysis of the church bell. As we do not have much knowledge on the dynamic of the bell, it is quite hard for us to dive straight into the modal analysis. Therefore, the profile of the church bell based on modal analysis requires further study. From the 3-D model developed using STRAND 7, a force stimulating an impact will be applied onto the bell. Different mode shapes will have different frequency; which will be compared with the natural frequency using analytical and numerical method. Upon completion of the impact and the modal analysis of the church bell, we will have a good knowledge of the dynamics of the English church bell

6 Reference

Biran, Adrian. 1999. *MATLAB 5 for engineering*: Harlow, England

Blakeborough, A. 2001. 'An analytical response of church bells to earthquake excitation'. *Journal of Earthquake Engineering*. 5(1): 69-92.

Brelyay, H., and Rossing, T.D. 1991 'Vibrational-modes of a noncircular bell with a church bell profile'. *Journal of the Acoustical Society of America* 90: 2196-2198.

Daggy, James. 1998. 'Bells and their music'. [http://www.msu.edu/~carillon/batmbook/chapter1.htm]. Accessed 13 Janunary 2004.

Fletcher, NH, McGee, WT, Tarnopolsky, AZ. 2002. 'Bell clapper impact dynamics and the voicing of a carillon'. *Journal of the Acoustical Society of America* 111: 437-1444.

Hibbert, Bill 2003. 'How bells make their sound'. [http://www.hibberts.co.uk/ears.htm]. Accessed 28 August 2003.

Kidd, Richard, and Fogg, Stuart. 2002 'A Simple Formula for the Large-Angle Pendulum Period'. *The Physic Teache.* 40: 81-83.

Lee, JM, Kim, SH, Lee, SJ, Jeong, JD, Choi, HG. 2002. 'A Study On The Vibration Characteristics Of A Larger Size Korean Bell'. *Journal of Sound and Vibration* 257(4): 779-790.

Marchand, Patrick. 1996. *Graphics and GUIs with MATLAB*: Boca Raton, Florida

Mathews, John. 1999. *Numerical method using MATLAB*: Upper Saddle River, N.J

Redfern, Darren. 1998. *The MATLAB 5 Handbook*: Springer, New York

Rossing, T.D. 1984. *Acoustic of Bells*: Van Nostrand Reinhold, New York

Terhardt, Ernst. 2002. ‘Strike note of bells’.
[http://www.mmk.ei.tum.de/persons/ter.html]. Accessed 04 March 2004.

Thomson, William. 2001. *Theory of vibration with application*: Cheltenham, England

7 MATLAB program

Mathlab coding for simple pendulum

```
function varargout = simple_gui(varargin)
% SIMPLE_GUI M-file for simple_gui.fig
%      SIMPLE_GUI, by itself, creates a new SIMPLE_GUI or raises the existing
%      singleton*.
%
%      H = SIMPLE_GUI returns the handle to a new SIMPLE_GUI or the handle to
%      the existing singleton*.
%
%      SIMPLE_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SIMPLE_GUI.M with the given input arguments.
%
%      SIMPLE_GUI('Property','Value',...) creates a new SIMPLE_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before simple_gui_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to simple_gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help simple_gui

% Last Modified by GUIDE v2.5 29-May-2004 16:30:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @simple_gui_OpeningFcn, ...
                   'gui_OutputFcn',  @simple_gui_OutputFcn, ...
```

```
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before simple_gui is made visible.
function simple_gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to simple_gui (see VARARGIN)

% Choose default command line output for simple_gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

%--------------------------------------------

axes(handles.axes2)
piccy = imread('pendulum.jpg');
image('CData', piccy);
set(gca,'XTickLabel',{[]})
set(gca,'YTickLabel',{[]})

%--------------------------------------------
```

```
% UIWAIT makes simple_gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = simple_gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global length
length= str2num(get(handles.edit1, 'string'));
tspan = [0 5];
data = getappdata(gcbf,'metricdata');
angle=data.edit5;
radian=angle*pi/180;
w0=[radian;0];
[t,w]=ode45(@pendulum,tspan,w0);

%Calculate frequency
frequency=(1/(2*pi))*(sqrt(9.81/length));
set(handles.text2, 'string', frequency);

%Calculate period
if angle<=10
   k=(((angle-0)/(10-0))*(1.574-1.571))+1.571;
```

```
elseif angle<=20
       k=(((angle-10)/(20-10))*(1.583-1.574))+1.574;
   elseif angle<=30
          k=(((angle-20)/(30-20))*(1.598-1.583))+1.583;
       elseif angle<=60
             k=(((angle-30)/(60-30))*(1.686-1.598))+1.598;
          elseif angle<=90
                k=(((angle-60)/(90-60))*(1.854-1.686))+1.686;
             elseif angle<=120
                   k=(((angle-90)/(120-90))*(2.157-1.854))+2.157;
                elseif angle<=150
                      k=(((angle-120)/(150-120))*(2.768-2.157))+2.768;
                   else angle>150
                      errordlg('K constant become infinate','Error');
                   end

period=((2*k)/pi)*((2*pi)*(sqrt(length/9.81)));
set(handles.text4,'string',period);

% figure;
a = get(handles.radiobutton1, 'value');
b = get(handles.radiobutton2, 'value');
c = get(handles.radiobutton3, 'value');
figure
if a == 1
   plot(t,w(:,1)*180/pi)
   ylabel('angle(degree)'),xlabel('time(s)')
elseif b ==1
   plot(t,w(:,2))
   ylabel('velocity(rad/s)'),xlabel('time(s)')
elseif c == 1
   plot(w(:,2),w(:,1))
   ylabel('angle velocity(rad/s)'),xlabel('angle(rad)')
else
   errordlg('No graph is selected','Error');
end
```

```
%--------------------------------------------------------------------------
function wd=pendulum(t,w);
global length
g=9.81;
[x,y]=size(w);
wd=zeros(x,y);%allocate space for wd
wd(1)=w(2);
wd(2)=-g/length*sin(w(1));

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
close;

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
```

```
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double
edit5 = str2double(get(hObject, 'String'));
if isnan(edit5)
   set(hObject, 'String', 0);
   errordlg('Input angle must be a number','Error');
end
data = getappdata(gcbf, 'metricdata');
data.edit5 = edit5;
setappdata(gcbf, 'metricdata', data);



% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of radiobutton1


% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton2


% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton3
```

Matlab coding for simple double pendulum

```
function varargout = simpledouble_gui(varargin)
% SIMPLEDOUBLE_GUI M-file for simpledouble_gui.fig
%      SIMPLEDOUBLE_GUI, by itself, creates a new SIMPLEDOUBLE_GUI or raises the existing
%      singleton*.
%
%      H = SIMPLEDOUBLE_GUI returns the handle to a new SIMPLEDOUBLE_GUI or the handle to
%      the existing singleton*.
%
%      SIMPLEDOUBLE_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SIMPLEDOUBLE_GUI.M with the given input arguments.
%
%      SIMPLEDOUBLE_GUI('Property','Value',...) creates a new SIMPLEDOUBLE_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before simpledouble_gui_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to simpledouble_gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help simpledouble_gui

% Last Modified by GUIDE v2.5 29-May-2004 17:03:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @simpledouble_gui_OpeningFcn, ...
                   'gui_OutputFcn',  @simpledouble_gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
```

```
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before simpledouble_gui is made visible.
function simpledouble_gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to simpledouble_gui (see VARARGIN)

% Choose default command line output for simpledouble_gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

%---------------------------------------------

axes(handles.axes2)
piccy = imread('pendulum1.jpg');
image('CData', piccy);
set(gca,'XTickLabel',{[]})
set(gca,'YTickLabel',{[]})

%---------------------------------------------

% UIWAIT makes simpledouble_gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = simpledouble_gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global length_1 length_2 mass_1 mass_2;

length_1= str2num(get(handles.edit1, 'string'));
length_2= str2num(get(handles.edit2, 'string'));
mass_1= str2num(get(handles.edit3, 'string'));
mass_2= str2num(get(handles.edit4, 'string'));

%seting the limits of the function
tspan = [0 40];

%initial condition theta(0)=1 radian,theat dot(0)=0 rad/s
data = getappdata(gcbf,'metricdata');
angle=data.edit5;
radian=angle*pi/180;
w0=[radian;0];
[t,w]=ode45(@pendulum,tspan,w0);
frequency=sqrt((9.81/(mass_2-mass_1))/((mass_2*length_2)-(mass_1*length_1)));
set(handles.text2, 'string', frequency);
```

```
%Calculate period
if angle<=10
   k=(((angle-0)/(10-0))*(1.574-1.571))+1.571;
elseif angle<=20
      k=(((angle-10)/(20-10))*(1.583-1.574))+1.574;
   elseif angle<=30
         k=(((angle-20)/(30-20))*(1.598-1.583))+1.583;
      elseif angle<=60
            k=(((angle-30)/(60-30))*(1.686-1.598))+1.598;
         elseif angle<=90
               k=(((angle-60)/(90-60))*(1.854-1.686))+1.686;
            elseif angle<=120
                  k=(((angle-90)/(120-90))*(2.157-1.854))+2.157;
               elseif angle<=150
                     k=(((angle-120)/(150-120))*(2.768-2.157))+2.768;
                  else angle>150
                     errordlg('K constant become infinate','Error');
                  end

period=((2*k)/pi)*(2*pi*sqrt(((mass_2*length_2)-(mass_1*length_1))/(9.81/(mass_2-mass_1))));
set(handles.text4,'string',period);


% figure;
a = get(handles.radiobutton1, 'value');
b = get(handles.radiobutton2, 'value');
c = get(handles.radiobutton3, 'value');
figure
if a == 1
   plot(t,w(:,1)*180/pi)
   ylabel('angle(degree)'),xlabel('time(s)')
elseif b ==1
   plot(t,w(:,2))
   ylabel('velocity(rad/s)'),xlabel('time(s)')
elseif c == 1
   plot(w(:,2),w(:,1))
   ylabel('angle velocity(rad/s)'),xlabel('angle(rad)')
```

```
else
   errordlg('No graph is selected','Error');
end

%--------------------------------------------------------------------

function wd=pendulum(t,w);
%This function describe the motion of the pendulum subjected to gravity in
%the form required by ODE23 and OD 45
%w(1)=angle theta
%w(2)=it's derivative
global length_1 length_2 mass_1 mass_2;
g=9.81;
[x,y]=size(w);
wd=zeros(x,y);%allocate space for wd
wd(1)=w(2);
wd(2)=g*(sin(w(1))*(mass_2-mass_1))/((mass_2*length_2)-(mass_1*length_1));

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
close;

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double
edit5 = str2double(get(hObject, 'String'));
if isnan(edit5)
   set(hObject, 'String', 0);
   errordlg('Input angle must be a number','Error');
end
data = getappdata(gcbf, 'metricdata');
data.edit5 = edit5;
setappdata(gcbf, 'metricdata', data);


% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1
```

```
% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton2


% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton3
```

Matlab coding for bell configuration

```
function varargout = bell_gui(varargin)
% BELL_GUI M-file for bell_gui.fig
%      BELL_GUI, by itself, creates a new BELL_GUI or raises the existing
%      singleton*.
%
%      H = BELL_GUI returns the handle to a new BELL_GUI or the handle to
%      the existing singleton*.
%
%      BELL_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in BELL_GUI.M with the given input arguments.
%
%      BELL_GUI('Property','Value',...) creates a new BELL_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before bell_gui_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to bell_gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help bell_gui

% Last Modified by GUIDE v2.5 29-May-2004 17:58:47

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @bell_gui_OpeningFcn, ...
                   'gui_OutputFcn',  @bell_gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
```

```
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before bell_gui is made visible.
function bell_gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to bell_gui (see VARARGIN)

% Choose default command line output for bell_gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

%---------------------------------------------

axes(handles.axes2)
piccy = imread('bell pendulum.jpg');
image('CData', piccy);
set(gca,'XTickLabel',{[]})
set(gca,'YTickLabel',{[]})

%---------------------------------------------

% UIWAIT makes bell_gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = bell_gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global theta ans l1 l2 l3 m1 m2
l1= str2num(get(handles.edit1, 'string'));
l2= str2num(get(handles.edit9, 'string'));
l3= str2num(get(handles.edit10, 'string'));
m1= str2num(get(handles.edit5, 'string'));
m2= str2num(get(handles.edit8, 'string'));

%seting the time limits of the function
tspan = [0 20];
time = [0 20];

data = getappdata(gcbf,'metricdata');
angle=data.edit6;
radian=angle*pi/180;
ang=data.edit7;
rad=ang*pi/180;
w0=[radian;0];
theta=[rad;0];
```

```
% % sign changing due to impact%%
% if (theta(:,1)*180/pi < -23)
%     theta(:,2)=theta(:,2)*-1;
%     w(:,2)=w(:,2)*-1;
% else (theta(:,1)*180/pi)==30
%     theta(:,2)=theta(:,2)*-1;
%     w(:,2)=w(:,2)*-1;
% end

[t,w]=ode45(@pendulum,tspan,w0);
[q,theta]=ode45(@clapper,time,theta);

%Calculate frequency
g=9.81;
value=(m2^2*l1*g*(l1+l3))+(m1*(l1+l2)^2)+...
(g*m2^2*(l1+l3)*l3^2)+(m1*(l1+l2)*m2*l3^2*g);

value2=6*m2^4*l1^3*g^2*l3+2*m2^2*l1^4*g*m1+...
2*m2^4*l1^3*g^2*l3^2+4*m2^4*l1^2*g^2*l3^3+9*m2^4*l1^2*g^2*l3^2+...
   2*m2^4*l1*g^2*l3^4-3*g^2*m2^4*l3^4*l1^2+2*g^2*m2^4*l3^5*l1-...
   4*m2^4*l3^2*l1^4*g^2-8*m2^4*l3^3*l1^3*g^2+4*m2^4*l3^3*l1*g^2+...
   2*m2^2*l1^2*g*m1*l2^2+2*m2^3*l1^3*g^2*m1*l3^2+2*m2^3*l1^2*g^2*m1*l3^2*l2+...
   4*m2^2*l1^2*g*l3*m1*l2+2*m2^2*l1*g*l3*m1*l2^2+2*m2^2*l1^3*g*l3*m1+...
   2*m2^3*l1^2*g^2*l3^3*m1+2*m2^3*l1*g^2*l3^3*m1*l2+4*m1*l1^2*l2*g*m2^2*l3^2+...
   4*m1*l1*l2*g*m2^2*l3^3+6*m1^2*l1^2*l2*m2*l3^2*g+6*m1^2*l1*l2^2*m2*l3^2*g+...
   2*m1*l2^2*g*m2^2*l3^2*l1+2*m1*l2^2*g*m2^2*l3^3+2*m1^2*l2^3*m2*l3^2*g+...
   2*m1*l1^3*g*m2^2*l3^2+2*m1*l1^2*g*m2^2*l3^3+2*m1^2*l1^3*m2*l3^2*g-...
   2*g^2*m2^3*l3^4*l1^2*m1-6*g^2*m2^3*l3^4*l1*m1*l2+2*g^2*m2^3*l3^5*m1*l1+...
   2*g^2*m2^3*l3^5*m1*l2+m1^2*m2^2*l3^4*g^2*l1^2+2*m1^2*m2^2*l3^4*g^2*l1*l2+...
   m1^2*m2^2*l3^4*g^2*l2^2-12*m2^3*l3^2*l1^3*g^2*m1*l2-...
   12*m2^3*l3^3*l1^3*g^2*m1-20*m2^3*l3^3*l1^2*g^2*m1*l2-...
   12*m1^2*m2^2*l3^2*l1^3*l2*g^2-12*m1^2*m2^2*l3^2*l1^2*l2^2*g^2-...
12*m1^2*m2^2*l3^3*l1^2*l2*g^2-12*m1^2*m2^2*l3^3*l1*l2^2*g^2-...
4*m1*m2^3*l3^2*l2^2*g^2*l1^2-8*m1*m2^3*l3^3*l2^2*g^2*l1-...
4*m1*m2^3*l3^4*l2^2*g^2-4*m1^2*m2^2*l3^2*l2^3*g^2*l1-...
4*m1^2*m2^2*l3^3*l2^3*g^2-4*m1^2*m2^2*l3^2*l1^4*g^2-...
```

```
4*m1^2*m2^2*l3^3*l1^3*g^2+4*m2^3*l3*l1^3*g^2*m1+...
4*m2^3*l3*l1^2*g^2*m1*l2+4*m2^3*l3^2*l1^2*g^2*m1+...
4*m2^3*l3^2*l1*g^2*m1*l2-8*m2^3*l3^2*l1^4*g^2*m1+...
4*m2^2*l1^3*g*m1*l2+m1^2*l2^4+m1^2*l1^4+m2^4*l1^4*g^2+...
6*m1^2*l1^2*l2^2+4*m1^2*l1*l2^3+4*m1^2*l1^3*l2+...
g^2*m2^4*l3^6;

frequency1=sqrt(((value+value2)^(1/2))/(2*m2^2*l3^2*l1^2+2*m1*(l1+...
   l2)^2*m2*l3^2-2*m2^2*l3*l1));
frequency2=sqrt(((value-value2)^(1/2))/(2*m2^2*l3^2*l1^2+2*m1*(l1+...
   l2)^2*m2*l3^2-2*m2^2*l3*l1));
set(handles.text16, 'string', frequency1);
set(handles.text4,'string',frequency2);

% figure;
a = get(handles.radiobutton1, 'value');
b = get(handles.radiobutton2, 'value');
c = get(handles.radiobutton3, 'value');
figure
if a == 1
   plot(t,w(:,1)*180/pi,'--')
   hold on;
   plot(q,theta(:,1)*180/pi)
   hold off;
   ylabel('angle(degree)'),xlabel('time(s)')
elseif b ==1
   plot(t,w(:,2),'--')
   hold on;
   plot(q,theta(:,2))
   hold off;
   ylabel('velocity(rad/s)'),xlabel('time(s)')
elseif c == 1
   plot(w(:,2),w(:,1),'--')
   hold on;
   plot(theta(:,2),theta(:,1))
   hold off;
   ylabel('angle velocity(rad/s)'),xlabel('angle(rad)')
```

```
else
   errordlg('No graph is selected','Error');
end
%--------------------------------------------------------------------------
function wd=pendulum(t,w);
global theta ans l1 l2 l3 m1 m2

g=9.81;
[x,y]=size(w);
wd=zeros(x,y);%allocate space for wd
wd(1)=w(2);
wd(2)=(((m2*g*(l1+l3)*sin(theta(1)))/l3)-(m1*g*(l1+l2)*sin(w(1)))-...
   (m2*g*(l1+l3)*sin(w(1))))/((m1*(l1+l2)^2)+(m2*l1^2)-((m2*l1)/l3));
ans=wd(2);

%--------------------------------------------------------------------------
function td=clapper(q,theta);

global ans l1 l2 l3 m1 m2

g=9.81;
[a,b]=size(theta);
td=zeros(a,b);%allocate space for wd
td(1)=theta(2);
td(2)=((-m2*l1*ans)-(m2*g*(l1+l3)*sin(theta(1))))/(m2*l3^2);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
close;

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%-------------------------------------------------------------------------
edit1 = str2double(get(hObject, 'String'));
if isnan(edit1)
   set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
data = getappdata(gcbf, 'metricdata');
data.edit1 = edit1;
setappdata(gcbf, 'metricdata', data);

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%---------------------------------------------------------------------------
edit2 = str2double(get(hObject, 'String'));
if isnan(edit2)
   set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
data = getappdata(gcbf, 'metricdata');
data.edit2 = edit2;
setappdata(gcbf, 'metricdata', data);

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%---------------------------------------------------------------------------
edit3 = str2double(get(hObject, 'String'));
if isnan(edit3)
   set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
data = getappdata(gcbf, 'metricdata');
data.edit3 = edit3;
setappdata(gcbf, 'metricdata', data);

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
%---------------------------------------------------------------------
edit4 = str2double(get(hObject, 'String'));
if isnan(edit4)
   set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
data = getappdata(gcbf, 'metricdata');
data.edit4 = edit4;
setappdata(gcbf, 'metricdata', data);

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%---------------------------------------------------------------------
edit5 = str2double(get(hObject, 'String'));
```

```
if isnan(edit5)
   set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
data = getappdata(gcbf, 'metricdata');
data.edit5 = edit5;
setappdata(gcbf, 'metricdata', data);

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%--------------------------------------------------------------------------
edit6 = str2double(get(hObject, 'String'));
if isnan(edit6)
   set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
```

```
data = getappdata(gcbf, 'metricdata');
data.edit6 = edit6;
setappdata(gcbf, 'metricdata', data);


% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%--------------------------------------------------------------------------
edit7 = str2double(get(hObject, 'String'));
if isnan(edit7)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
data = getappdata(gcbf, 'metricdata');
data.edit7 = edit7;
```

```
setappdata(gcbf, 'metricdata', data);

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%---------------------------------------------------------------------
edit8 = str2double(get(hObject, 'String'));
if isnan(edit8)
   set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
data = getappdata(gcbf, 'metricdata');
data.edit8 = edit8;
setappdata(gcbf, 'metricdata', data);

% Hints: get(hObject,'String') returns contents of edit8 as text
```

```
%       str2double(get(hObject,'String')) returns contents of edit8 as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%---------------------------------------------------------------------
edit9 = str2double(get(hObject, 'String'));
if isnan(edit9)
   set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
data = getappdata(gcbf, 'metricdata');
data.edit9 = edit9;
setappdata(gcbf, 'metricdata', data);

% Hints: get(hObject,'String') returns contents of edit9 as text
%       str2double(get(hObject,'String')) returns contents of edit9 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%--------------------------------------------------------------------------
edit10 = str2double(get(hObject, 'String'));
if isnan(edit10)
   set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
data = getappdata(gcbf, 'metricdata');
data.edit10 = edit10;
setappdata(gcbf, 'metricdata', data);

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double


% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1


% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton2


% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton3
```