Fall 12-2012

Finding Unpredictable Behaviors of Periodic Bouncing for Forced Nonlinear Spring Systems when Oscillating Time is Large

Yanyue Ning
University of Connecticut - Storrs, ivyyning@163.com

Finding Unpredictable Behaviors of Periodic Bouncing for Forced Nonlinear Spring Systems when Oscillating Time is Large

Yanyue Ning

A Thesis

Submitted in Partial Fulfillment of the

Requirements for Graduation as

University Scholar and Honors Scholar

At the University of Connecticut

Storrs

2012

Approval Page

Honors Scholar Thesis

Finding Unpredictable Behaviors of Periodic Bouncing for Forced Nonlinear Spring Systems when Oscillating Time is Large

Presented by

Yanyue Ning

Research Advisor _____

Patrick J. McKenna

Honors Advisor _____

David R. Solomon

Department of Mathematics

College of Liberal Arts and Sciences

University of Connecticut

2012

Acknowledgements

I would like to express my deepest gratitude to Prof. Joseph McKenna who guided my research in the fall semester of 2012. His research concentration in the qualitative study of nonlinear partial differential equations intrigues me a lot, and his idea that applying mathematics to solve mechanical problems of bridges deepens my interest in the field of computational mathematics and engineering. Moreover, his passion in applied mathematics passed on to me throughout the research and we even threw ourselves into the research with enthusiasm during the thanksgiving break. From his advice of technical skills and work habits, I felt how important that seriousness and meticulous nature are for a competent scientific worker and I will always remember his sincere teachings in my future study and career.

I am also grateful to thank Dr. Reed Solomon, who is the honors advisor for my two years study in the honors program. Before starting this research, he matched my interest with each mathematics professor's research field in order to find the best combination, which helped me to set the stage for the thesis and get the best training of research skills.

Though this honors thesis is the final work of my undergraduate study, I believe it will be one of accomplishments in my life. On the occasion of graduation, I will also express my great gratitude to Prof. Sarah Glaz who provides moral encouragement and support for me since my sophomore year, which makes me feel more confident in this research and my learning in applied mathematics. Also, I would like to thank Prof. Dmitriy Leykekhman, who provided me with very useful help of MATLAB coding during the research process.

Lastly, I am thankful to my parents and friends for all the helps and encouragement they always provide.

Table of Contents

I. Abstract.

The model of nonlinear spring systems can be applied to deal with different aspect of mechanical problems, such as oscillations in periodic flexing in bridges and ships. The concentration of this research is the bouncing behaviors of nonlinear spring system when the processing time is large, therefore nonlinear ordinary differential equations (ODE) are suitable since researchers can add different variables into the models and solve them by computational methods. Benefit from this, it is easy to check the oscillations or bouncing behaviors that each variable contributes to the model and find the relationship between some important factors: vibrating frequency, external forces and amplitudes. Moreover, analyzing the model can be implemented by plugging different values into the equations characteristics. For example, this research will focus on discussing various initial conditions since they may cause different behaviors to appear. Conducting numerical analysis to check the performance of the model by computing with MATLAB is also necessary during the research procedure, which may help researches to avoid failure results and show the existence of a certain phenomenon.

II. Introduction.

When we suspend a single particle on a linear spring and put damping and external periodic forces on the spring, the solution can be found by a specific initial condition and external periodic forcing. There will be no effect on the long-term behaviors when we plug in large initial values. However, the result we get could be qualitatively different when we do research for nonlinear situations. In the nonlinear spring systems, the long-term behaviors will depend on both the external forcing and the initial conditions. For example, a very small periodic force can not only cause the small long-term consequences like that of linear models but can also show us unexpected large periodic oscillations with different initial values.

Before creating the nonlinear differential equations model, we have to analyze the spring constant contained in the Hooke's Law since this factor can affect the model and even the bouncing behaviors. Then we can simulate the behaviors of vibrating of a linear spring with different elasticity constant by using MATLAB. However, the most important thing we care about in this research is the bouncing behavior when the time is very large, say from 6000 to 9000, and with different initial values. We look forward to seeing what can happen when we take absolute value for the solution of the differential equation and plug in a large time period with different initial conditions. Moreover, when changing the frequency, the magnitude of external forces and damping coefficients, the bouncing behaviors can have corresponding variations. Another task of this research is to use 3-D graph to show the relationship for vibrating frequency, magnitude of external forces and the range of amplitudes. We will also use other MATLAB programming results to present new details on how increasing initial values affects the response of the system. In this presentation, we will also provide numerical evidence to explain the

theoretical results, which can clearly shows the relationships between each parameter in this model.

III. Finding Periodic Oscillation Solutions when Time is Large

A physical problem occurs when we put a particle attached to a spring, and the second order ODEs can fully manifest the oscillations of a spring system. For example, choose a nonlinear unforced equation model $y''+ py'+ qy^3 = 0$, where p is damping constant and q is spring constant. We set $p = 0.1$ and $q = 1$, the equation becomes $y''+ 0.1y'+ y^3 = 0$. Let set initial values to be $y(0) = 20$ and $y'(0) = 0$, and make the oscillating time to be 0 to 50, we can get the graph

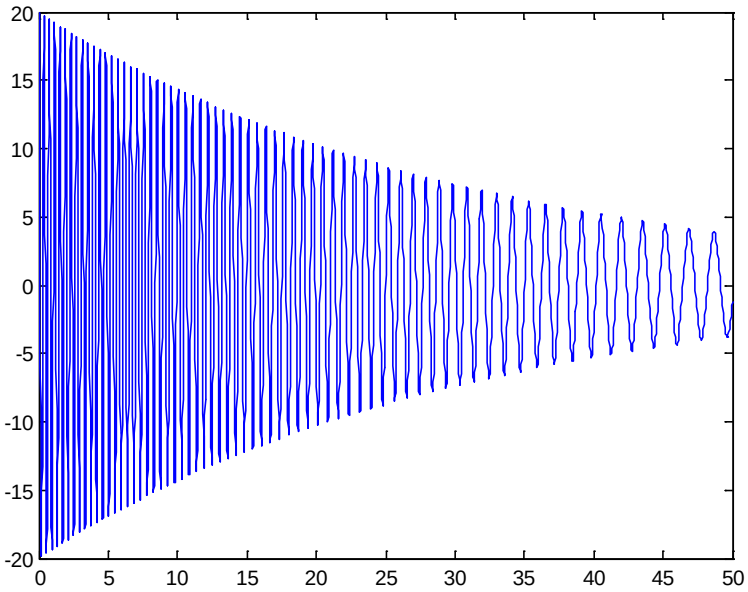

Figure 1

Here we can see that since we put a damping term in the model, so the graph will be convergent to the equilibrium position. As the time period become much larger, the oscillation will eventually disappear. For example, when we set the choose the time interval to be 0 to 200, the

graph almost shows a straight line from 160 to 200, which means few or no oscillations in this time period

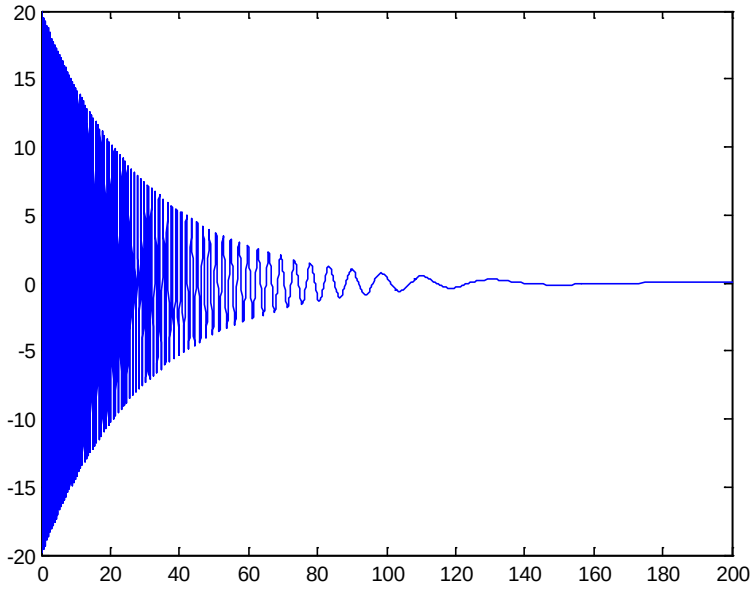

Figure 2

An important task we have to do before modeling the oscillations is to add a factor of nonlinearity to the model. Basically, we can discuss the relationship between two different spring constants to deal with this problem. Consider the case that a mass is hanged by both a linear spring and a rubber band, and each of them has an elastic constant. In this situation, the particle moving in $-\infty < x < \infty$ is determined by the two restoring forces: by when y is positive, and ay when y is negative. We can create a system of functions to describe the nonlinear relationship:

$f(y) = \begin{cases} by, & y > 0 \\ ay, & y < 0 \end{cases}$; consider the example $b = 17$ and $a = 13$, the graph is

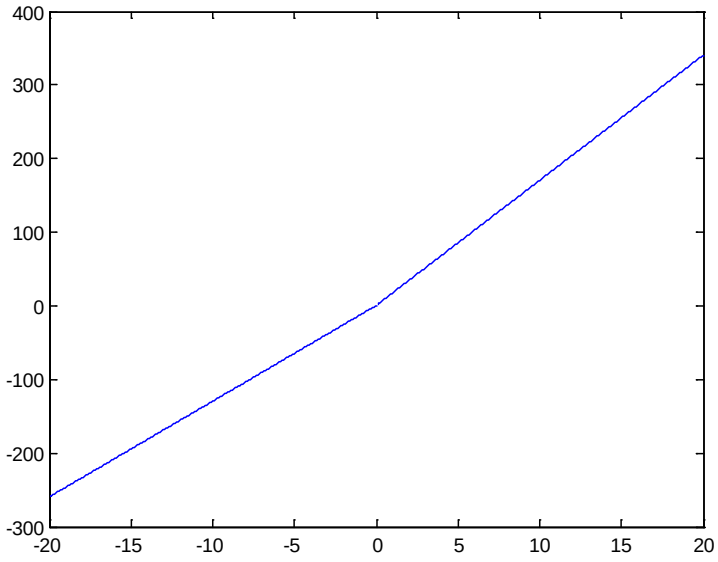

Figure 3 Nonlinearity

From the graph, we can see that $x = 0$ is a breaking point which separates the line into two segments and the right section has a larger slope than the left, but the nonlinearity is not strong. To get a stronger nonlinearity, one has to make a larger difference between a and b.

In order to apply this function system to our nonlinear ODE model to describe the oscillation of a spring, we need to describe the nonlinearity by using the combined form: $ay^+ - by^-$ ($a > b$) where $y^+ = y$ if $y > 0$ and 0 otherwise, and $y^- = -y$ if $y < 0$ and 0 otherwise. This expression can show the nonlinearity in the equation, and a large difference between a and b can show a strong nonlinearity as stated above. As what we discussed just now, a spring system with a small forcing term and a damping term may cause unexpected oscillating behaviors. For our first model, we set the forcing term to be $f(t) = \lambda \sin \mu t$, where λ is an external force applied on the spring and μ stands for the frequency of oscillation. Combined what we have indicated above, the nonlinear differential equation for this model is given by

$$y'' + 0.01y' + ay^+ - by^- = 10 + \lambda \sin \mu t$$

The same as previously shown, we still set $b=17$ and $a=13$. Let us first see what may happen when we put a small force and a large frequency into the model, say λ =0.1and μ =4, then the equation becomes $y''+0.01y'+17y^{+}-13y^{-}=10+0.1\sin 4t$. It would be hard to code $17y^{+}-13y^{-}$ directly in MATLAB since it cannot recognize this expression, but we can transform $17y^{+}-13y^{-}$ to $15y+2|y|$ in the coding process to avoid errors. In this research, we will focus on the oscillations when we make the time period to be very large, such as from 800 to 2500. In this situation, we can compare the graphs with three different initial values

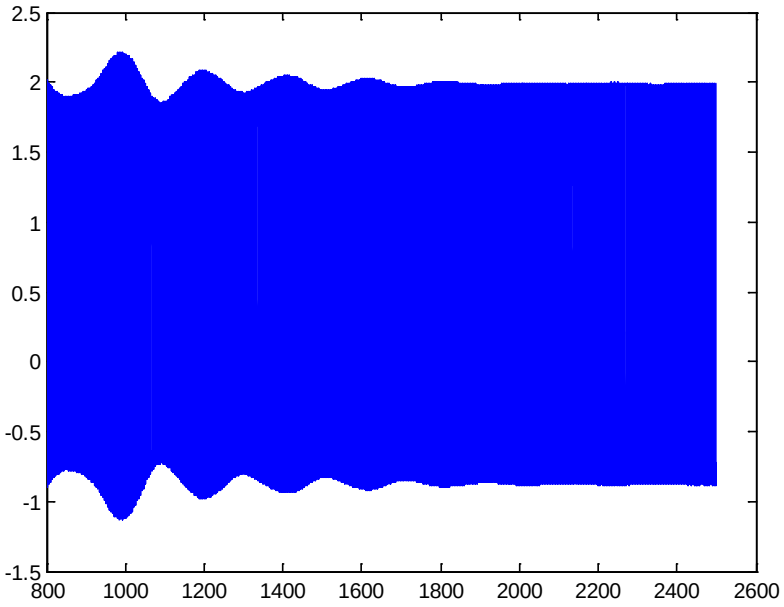

Figure 4.1 Initial conditions $y(0)=2, y'(0)=-1$

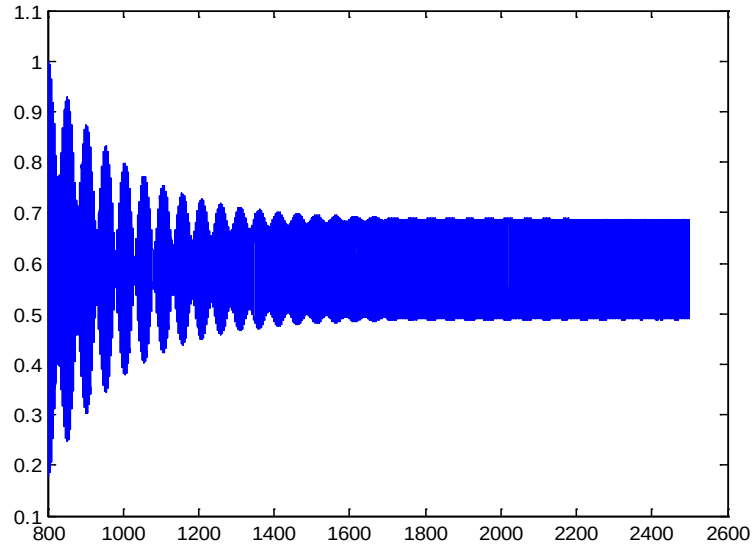

Figure 4.2 Initial conditions $y(0)=1, y'(0)=0$

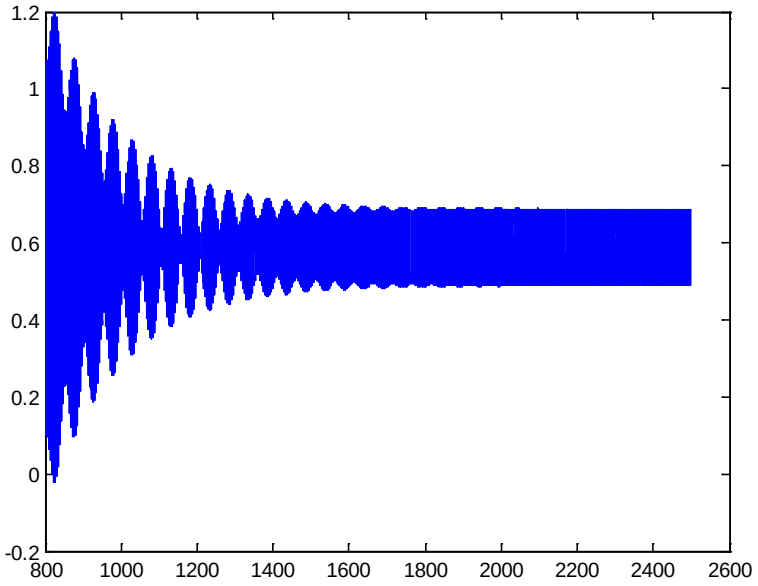

Figure 4.3 Initial conditions $y(0)=0.1, y'(0)=-0.1$

From the graphs of decreasing initial values, we can see that the amplitudes at point $t=800$ also decrease and the equilibrium position is some point near $y=0.5$. Moreover, we can see much faster shrinking amplitudes with smaller initial values, which is especially obvious for the period

of 800 to 1200; for Figure 4.1 with initial values of $y(0) = 2$ and $y'(0) = -1$, there is almost no shrinking behaviors. Another property we find out is that the time each oscillation needs to settle down to the eventual periodic state is almost the same and we can get periodic solutions after $t = 2000$.

One may also ask what may happen when we put a large force on the spring. This time we may assign a large value of λ and a small value of μ with a large difference between b and a. Consider respective values $b = 17$, $a = 1$, $\lambda = 13.7$ and $\mu = 0.17$, thus the equation becomes $y'' + 0.01y' + 17y^+ - y^- = 10 + 13.7\sin 0.17t$. Similarly, we can use $9y + 8|y|$ instead of $17y^+ - y^-$ in MATLAB programming. We try to run the codes in the period of 800 to 2000, and we get three different graphs with the same initial conditions of last example

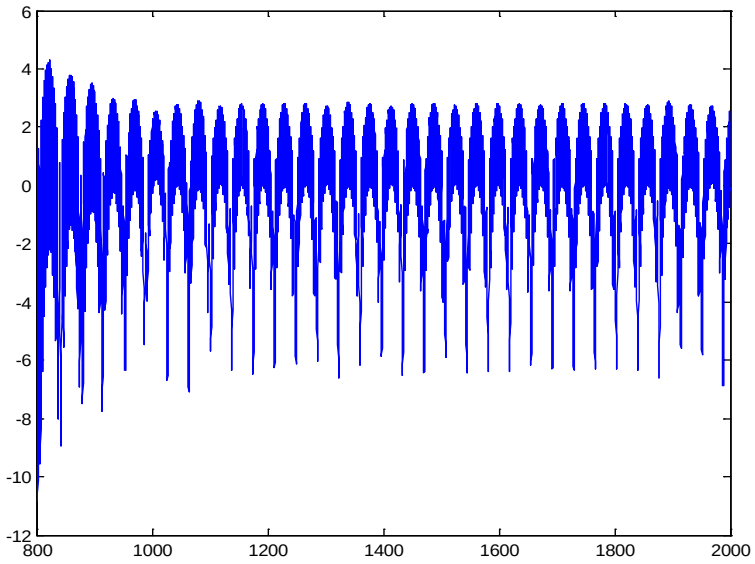

Figure 5.1 Initial conditions $y(0) = 2, y'(0) = -1$

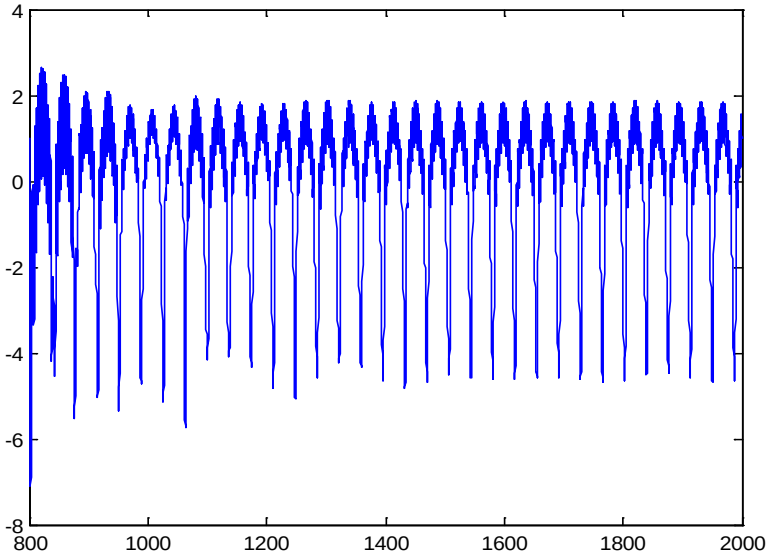

Figure 5.2 Initial conditions $y(0) = 1, y'(0) = 0$

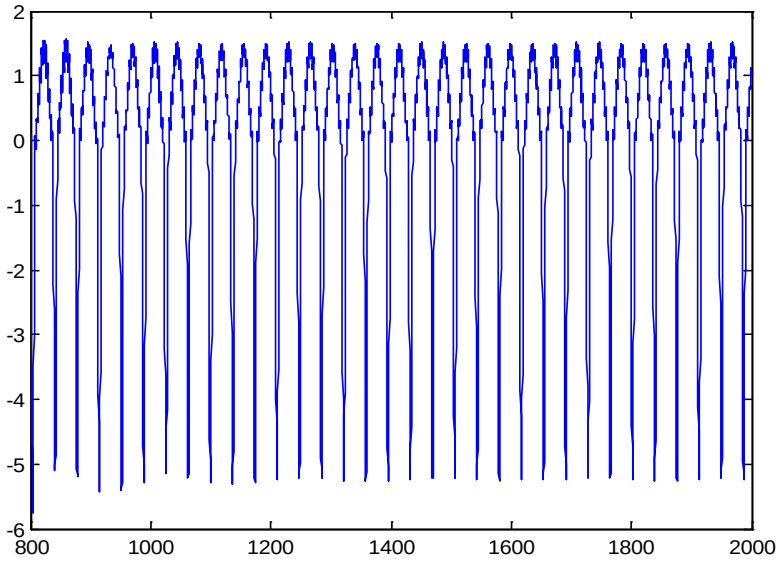

Figure 5.3 Initial conditions $y(0) = 0.1, y'(0) = -0.1$

From the results, when we make the nonlinearity become stronger, we can see many types of stable periodic solutions as the time period is very large, and a higher density of oscillations for each period is presented by choosing larger initial values. In addition, we also see that the largest amplitudes decrease as the initial values decrease.

IV. Finding Periodic Bouncing Behaviors when Time is Large

In general, we put the particle near the equilibrium position and the particle will follow the usual linear equation. Now, there is a new case that we add a small forcing term and a small viscous damping term to the spring system, and we would hope to see periodic bouncing motions. The basic model for the periodic bouncing solutions is given by the following set of equations

$$\begin{cases} u''+\delta u'+bu=\pm 1+\varepsilon \sin t \\ u\geq 0 \\ u(t_0)=0 \Rightarrow u'(t_0+)=-u'(t_0-) \end{cases}$$

and the general nonlinear differential equation is given by $y''+\varepsilon a(\varepsilon)y'+by=1+\varepsilon \sin(t-\alpha_\varepsilon)$, where $a(\varepsilon)\to 0$ as $\varepsilon \to 0$, $a(\varepsilon)>0$ and a is a continuously differentiable function. In this study, we will discuss the bouncing behaviors in a large time with a constantly small damping coefficient $p=0.01$ for the model

$$y''+0.01y'+by=sign(y)*(a+\lambda \cos \mu t)$$

Before finding the bouncing behaviors, we can first have a look at what the oscillation behavior of this model is when the time is large. Here, we are hoping to see the results in the interval from 2500 to 3000. Consider plugging in values that $b=a=\lambda=4$ and $\mu=0.1$, we can see that the oscillations is the following

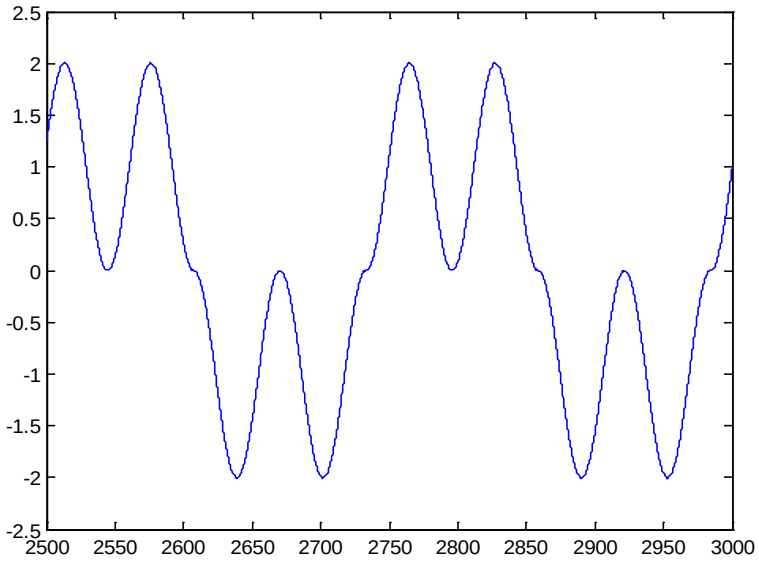

Figure 6 Oscillations of $y''+0.01y'+4y=sign(y)*(4+4\cos 0.1t)$

We can see from this graph that $y=0$ is the equilibrium position and the graph is almost symmetric on this line. Therefore, this is a good starting point for us to think about how to get the graph of bouncing. Just as its name "bouncing" implies, the amplitudes should be always positive, which means the value of all *y* should be nonnegative all the time. Here, we can use numerical technique to change one point to our code of Figure 6, namely, we can find the absolute values of *y* instead of original *y* in order to make the amplitudes nonnegative. Moreover, one may realize that the curve that ode45 graphs is stiff without polish. To solve this problem, we recommend using "`eps`" as the option to make the graph looks non-stiff. Since we want the period to be 2500 to 3000, the numerical methods we consider to correctly express this time interval is setting $timefinal=3000$ and create a new variable i which satisfies $i=\min(t>timefinal-500)$. This command can always require MATLAB to start drawing bouncing curves at $t=2500$ rather than $t=0$. After revising the code, MATLAB shows us perfect

bouncing behaviors from 2500 to 3000, and we can see that there is no weakening trend so the damping term does not affect the bouncing too much.

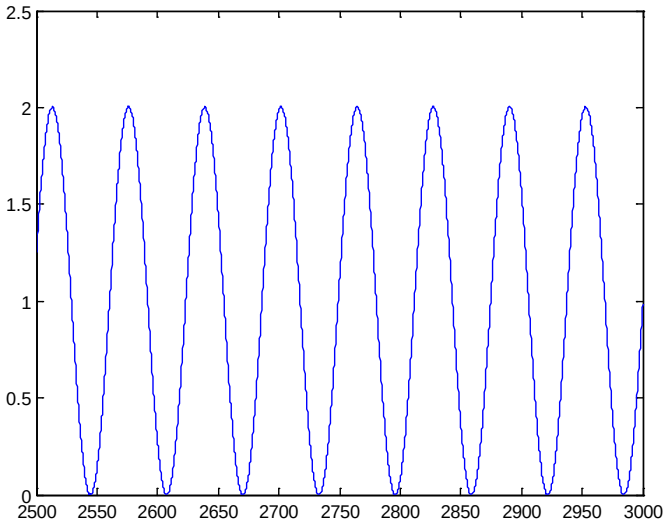

Figure 7 Bouncing Behaviors of $y"+0.01y'+4y=sign(y)*(4+4\cos 0.1t)$

Though the bouncing behaviors can appear, it does not occur at any time since those four variables a,b,λ,μ can affect the original oscillation. For example, when plugging in $a=20,b=20,\lambda=4,\mu=0.2$, the graph for oscillation is

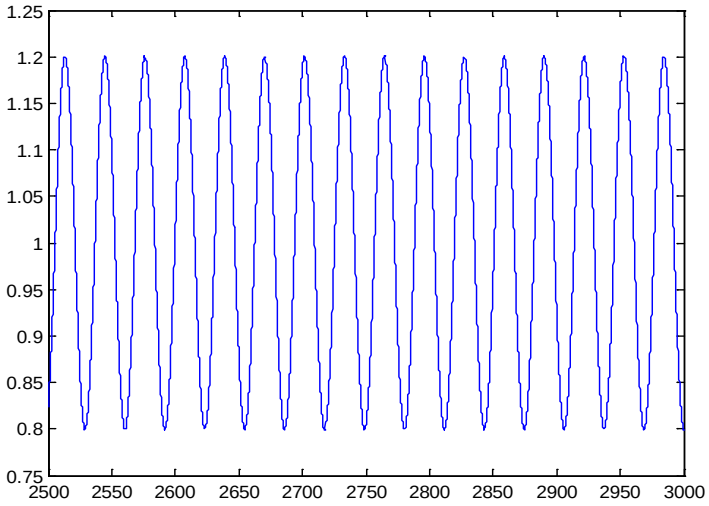

Figure 8 Oscillations of $y"+0.01y'+20y=sign(y)*(20+4\cos 0.2t)$

Even if we take absolute values for all y, there will be no bouncing behaviors occurring since the original oscillation solutions are already all positive so that the curve of bouncing will never touch $y = 0$. In other words, if we want a bouncing occurs, we should require some part of the original oscillation curve to intersect with the line of $y = 0$ but not necessarily to be symmetric. Numerically, to guarantee the bouncing appears, we have to find the variables which satisfy the requirements that the step size between each y value has to be very small (less than 0.003) and the set of y should contain both positive and negative values. Here are two subset of Y value for Figure 7 and 8, we see that the data in Figure 7 can satisfy above requirements but the data of Figure 8 cannot, so $a = 20, b = 20, \lambda = 4, \mu = 0.2$ cannot provide bouncing behaviors to the model.

Figure 7

-0.0055
-0.0044
-0.0035
-0.0027
-0.0021
-0.0016
-0.0013
-0.0010
-0.0007
-0.0005
-0.0003
-0.0001
0.0001
0.0003
0.0005
0.0007
0.0008
0.0010
0.0011
0.0013

Figure 8

0.8070
0.8064
0.8059
0.8052
0.8045
0.8038
0.8031
0.8025
0.8018
0.8013
0.8008
0.8004
0.8001
0.7999
0.7998
0.7997
0.7996
0.7996
0.7996

Therefore, one has to try different values for the four variables on this experiment several times to get bouncing behaviors. After several attempts, we find that the bouncing behaviors can occur successfully when we set $b = a = \lambda$ or $b \approx a \approx \lambda$, but for the second condition, the graphs may be manifested by some chaotic tiny-amplitude bouncing in addition to general bouncing behaviors. For example, when we choose $b = 10.2, a = 9.9, \lambda = 10$ and $\mu = 0.1$, we can see some small dense bouncing behaviors when the amplitudes are near $y = 0$ (which has been marked by the red box).

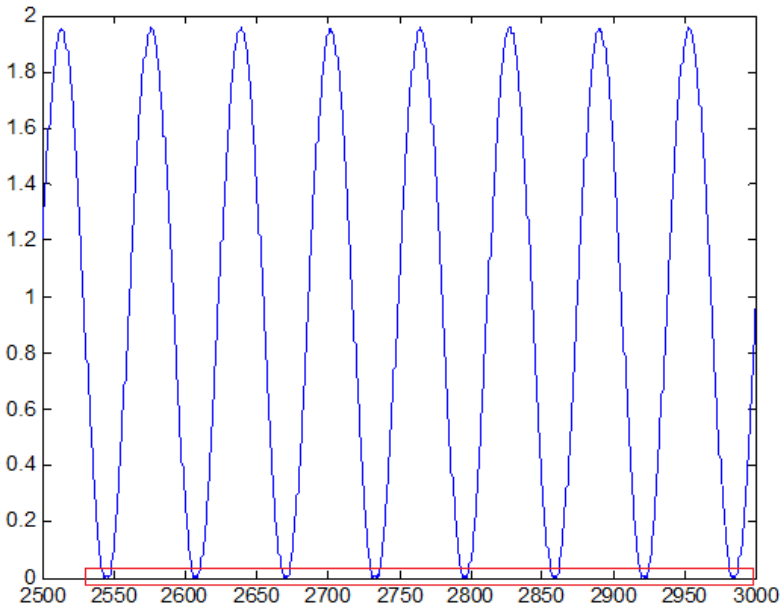

Figure 9 Bouncing Behaviors of $y'' + 0.01y' + 10.2y = sign(y) * (9.9 + 10\cos 0.1t)$

Now we enlarge the graph to see more details for these bouncing with small amplitudes, and there is no specific way to control and refine these solutions, so these are chaotic bouncing behaviors.

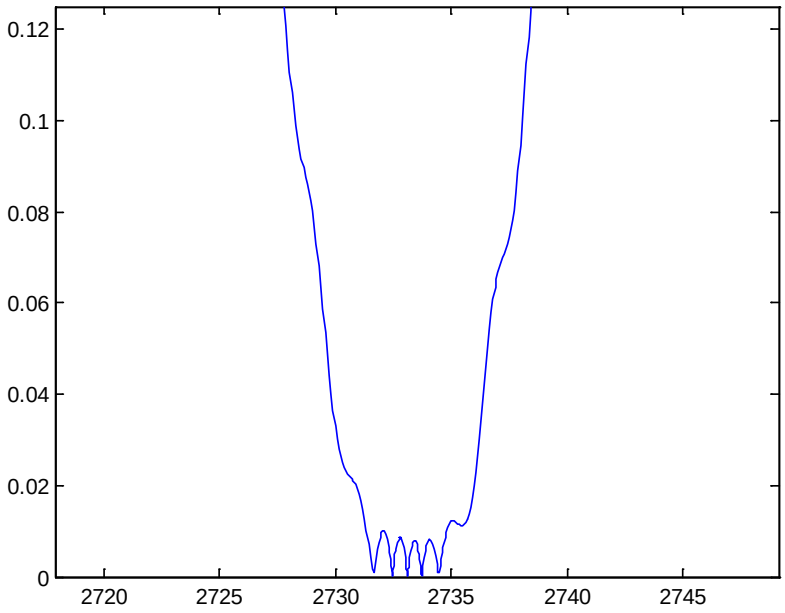

Figure 10 Enlarged Chaotic Behaviors for $y"+0.01y'+10.2y=\mathrm{s}ign(y)*(9.9+10\cos 0.1t)$

In addition, when we try different initial values to the graph, there is no too much difference between each of them. Consider the equation $y"+0.01y'+4y=\mathrm{s}ign(y)*(4+4\cos 0.1t)$, we try three different initial values and get three graphs which are almost the same

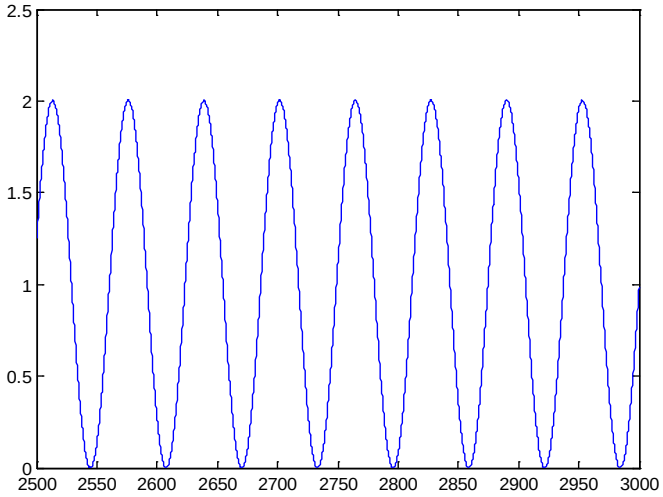

Figure 11.1 Initial conditions $y(0)=2, y'(0)=-1$

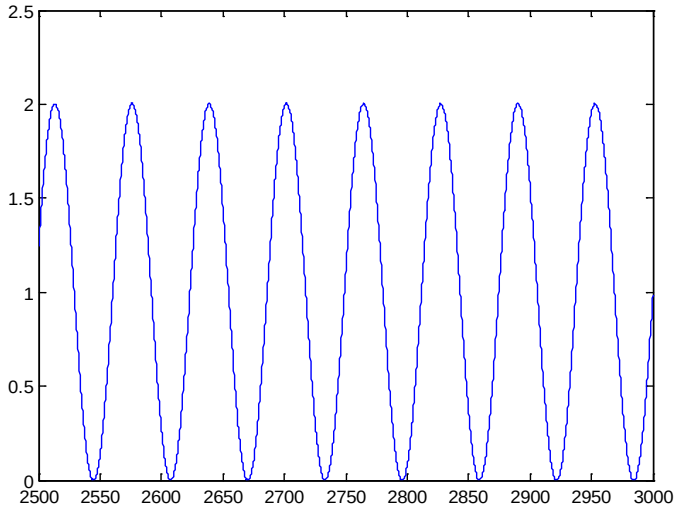

Figure 11.2 Initial conditions $y(0)=1, y'(0)=0$

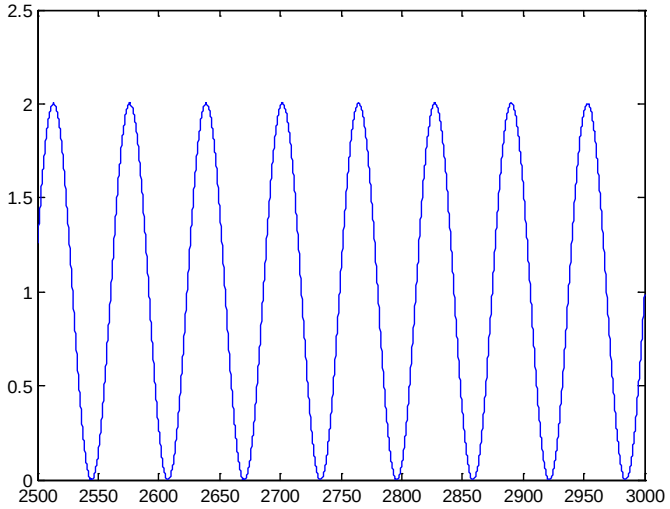

Figure 11.3 Initial conditions $y(0)=0.1, y'(0)=-0.1$

Furthermore, when set $a=b=10$ and keep λ and μ to be unknown, our model becomes $y''+0.01y'+10y=sign(y)*(10+\lambda\cos\mu t)$ and researchers may want to know the relationships for the bouncing frequency, force applied on the spring and the range of amplitudes when choosing different initial values. This is a good topic to research since we can use 3-D graph to see effects that both variational frequencies and forces on the range of amplitudes. According to

this new idea, λ and μ will no longer be constants and we have to put the variations of both factors into consideration when programming with MATLAB. Here, we can use "for loops" to describe their variations. For μ, we set the range to be 2.5 to 5 with the changing step size of 0.1, so there will be 25 steps in the whole process. For λ, we set the range to be much smaller, say 0 to 1, and we set the step size to be 0.05 so that there will be 20 steps in the process. It will take a long time to run the programs with different initial conditions, since when MATLAB reads one μ, it has to check the values of twenty λ s, and then to read the next μ with the same twenty λ s; the rest can be done in the same manner. Thus the graphing procedure will take a much longer time than previous graphs. In this topic, we are concerned about the range of amplitudes, which basically means difference between the maximum and minimum amplitudes; similarly, we also care about the behaviors when the time is very large, so we still set the running time to be 2500 to 3000. In order to easily compare the three graphs with different initial conditions, we can set same scale for z-axis of each graph, say 0 to 10. The results are the following

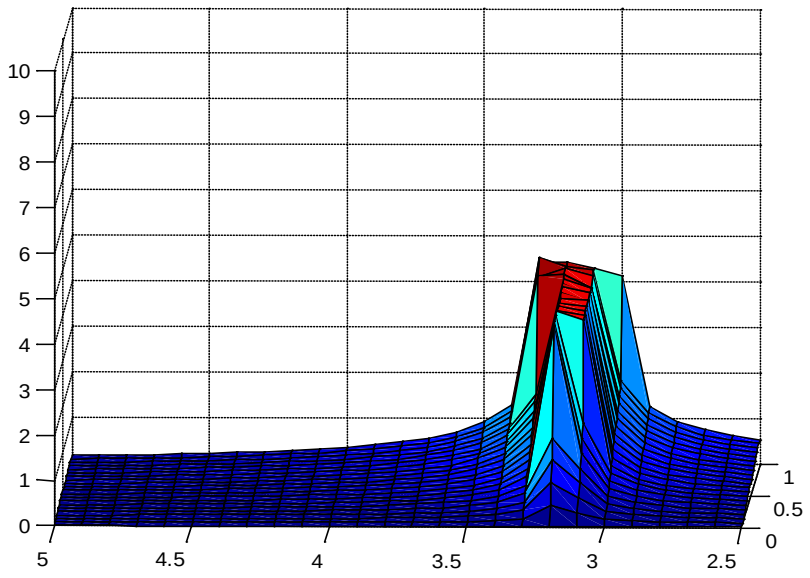

Figure 12.1 Initial conditions $y(0) = 0.1, y'(0) = -0.1$

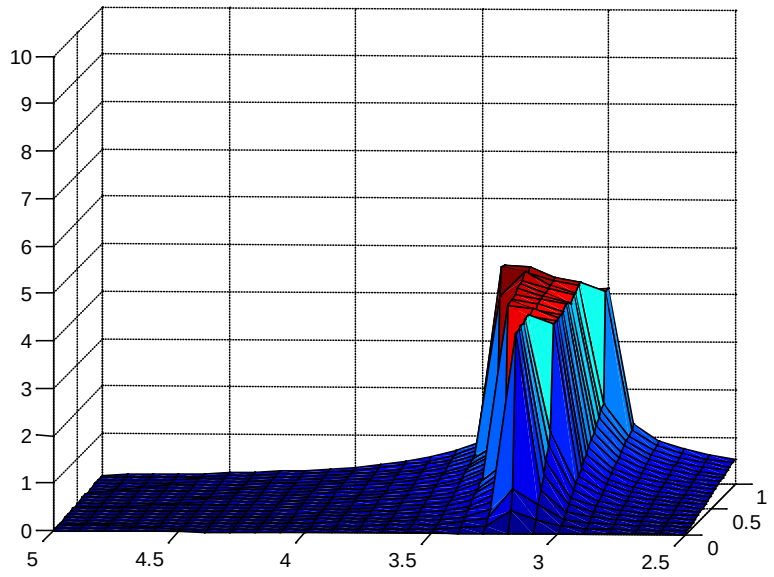

Figure 12.2 Initial conditions $y(0)=1, y'(0)=0$

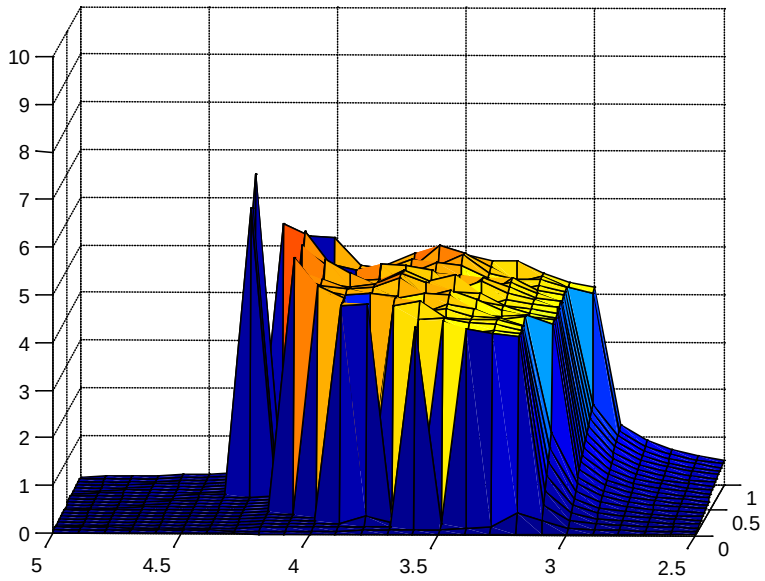

Figure 12.3 initial conditions $y(0)=2, y'(0)=-1$

Now let us analyze the three graphs. First, when the initial values become larger, the range of amplitudes (or say, the z-values) becomes larger, which has been shown especially obviously when shifting from Figure 12.2 to 12.3. Moreover, the value of range gives a larger

and faster jump for each corresponding pair of λ and μ when the initial values get larger and larger. To clearly show this statement, we choose the point $\mu = 3.2$ and see when we change the value of λ, the values of amplitude range for each initial condition is given by the following table

	λ=0	λ=0.05	λ=0.1	λ=0.15	λ=0.2	λ=0.25
Amplitude Range for Figure 12.1	0	0.4	0.8	1.2	1.6	4.1
Amplitude Range for Figure 12.2	0	0.4	1.1	4.1	4.3	4.3
Amplitude Range for Figure 12.3	0	0.35	4.1	4.1	4.2	4.2

This property can also been shown by λ sides of Figure 12

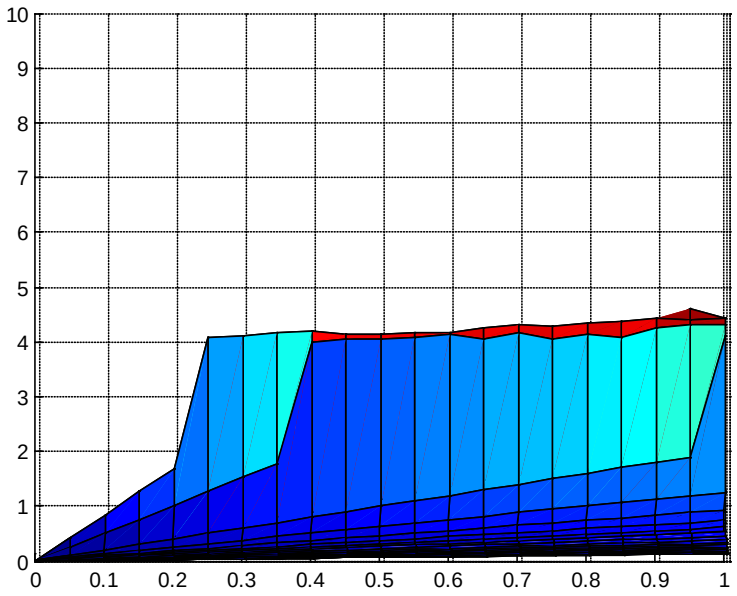

Figure 13.1 Initial conditions $y(0) = 0.1, y'(0) = -0.1$

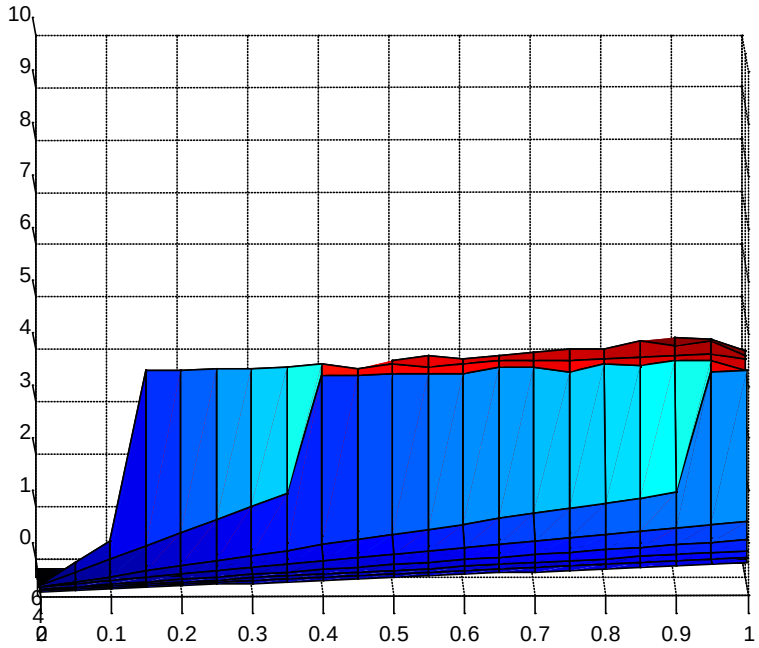

Figure 13.2 Initial conditions $y(0)=1, y'(0)=0$

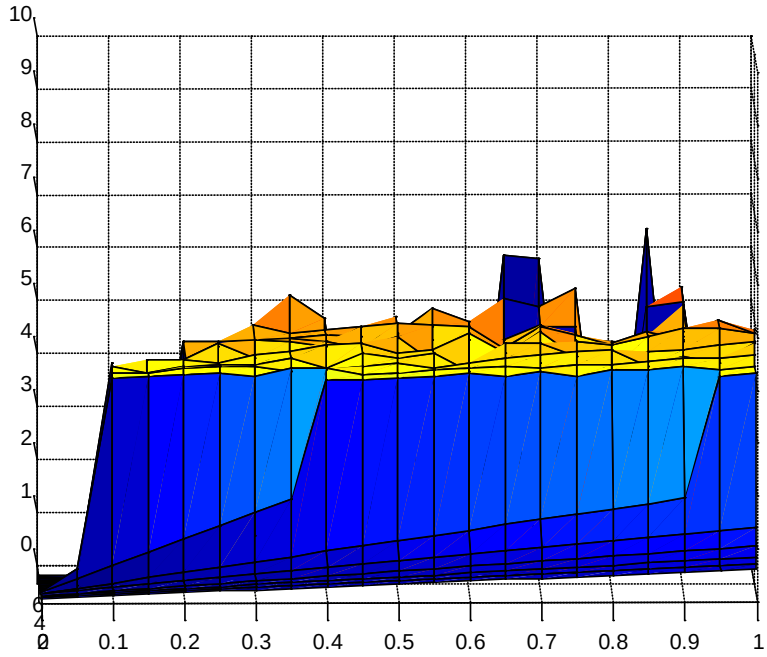

Figure 13.3 Initial conditions $y(0)=2, y'(0)=-1$

Another important finding from Figure 12 is that when the initial values become larger, a surface expands gradually on the top of each 3-D graph. Now the problem which brought to our attention is whether this area of the surface will expand and continue to exist when we plug into much larger initial values. To solve this problem, we can perform a numerical experiment to check the slice of the surface; if the slice will never suddenly change or even disappear, we will conclude that surface exists all the time. However, this experiment may require a set of more accurate initial values. Now, we set $\lambda = 0.5$ and the nonlinear equation becomes

$$y'' + 0.01y' + 10y = sign(y) * (10 + 0.5\cos \mu t)$$

To solve this equation, we randomly choose initial values, say $y(0) = 2$ and $y'(0) = -1$, and let μ initially be 3.1. Then set the time to be 0 to 3000, then we can solve the nonlinear ODE and get results for $y(3000)$ and $y'(3000)$. We can use this way to find a point in the 3-D space on each graph of Figure 12, and this point shows the magnitude of amplitude range for a specific μ. Yet this is just a single point, we still need more points and connect them to form a line which describes the slice of surface. Thus the next attempt when we solve the same function, we want $\mu = 3.0$ and set the time to be 3000 to 6000. For the initial values, we have a pair of accurate initial values which are the final values of our last attempt, i.e. $y(3000)$ and $y'(3000)$. The MATLAB codes for the first two steps are the following:

First Step:

```
function dy=function7(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-10*y(1)+sign(y(1))*(10+0.5*cos(3.1*t));

[T,Y]=ode45('function7',[0 3000],[0.1 -0.1],'AbsTol',eps,'NonNegative',eps);

Y(end,1)
ans = -1.4992
```

```
Y(end,2)
ans = 2.4617
```

Second Step:

```
function dy=function8(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-10*y(1)+sign(y(1))*(10+0.5*cos(3*t));

[T,Y]=ode45('function8',[3000 6000],[-1.4992
2.4617],'AbsTol',eps,'NonNegative',eps);

Y(end,1)
ans = 1.1061
Y(end,2)
ans = 1.4603
```

Similarly, the rest of steps can be done in the same manner. To summarize this experiment, we will explain this with numerical techniques. For μ, it starts at 3.1 and decrease by 0.1 for each step and it will end at 2.1, so totally there will be 10 steps. We also see that the time interval will increase by 3000 for each step, basically it will be 0 to 3000, then the next step be 3000 to 6000, and next continue with 6000 to 9000… Thus the current problem is how to connect the time interval and the values of μ. Here, we create a new variable, n, which starts from 0 and ends with 10. This variable stands for the number of steps. We can perform a "for loop" on n and use n to connect μ and time interval. Basically, $\mu = 3.1 - n \times 0.1$, $timeinitial = 3000 \times n$ and $timefinal = 3000 + timeinitial$. Since we have to use final values to be the initial values of the next step, we should set $inia$ and $inib$ to stands for two variational initial values. Based on what we discussed above, we can randomly assign values for them at the beginning, such as $inia = 2$ and $inib = -1$; inside the loop, we replace the expressions with $inia = Y(timefinal, 1)$ and $inib = Y(timefinal, 2)$.

We hope see that the surface expands gradually when the initial value becomes larger, if this actually occurs, there could exists a turning point on the curve. Namely, before the turning

point, the z-values decrease when μ decreases; after the turning point, the z-values decrease when μ increases. Now, let us see what happens in reality:

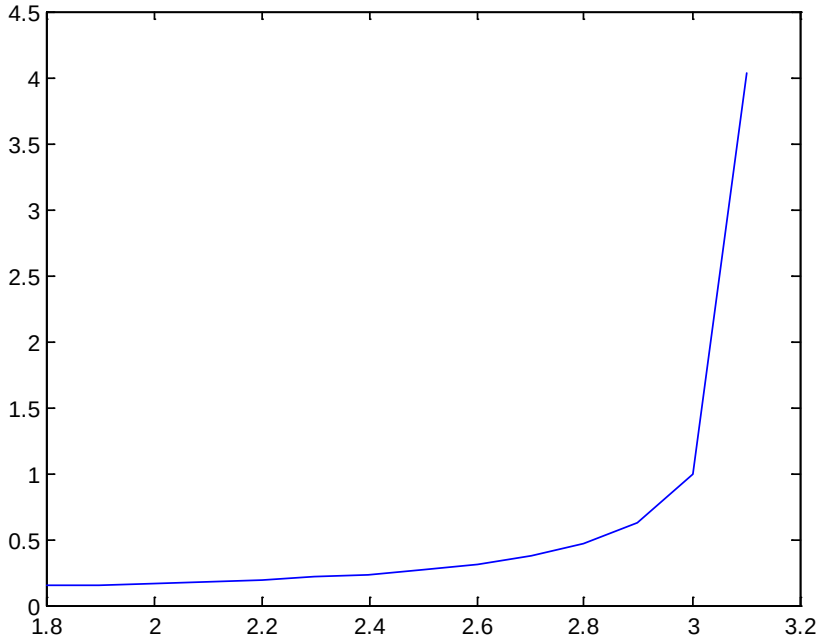

Figure 14 Decreasing μ with step size of 0.1

We can see that there is no turning point appearing on the graph, instead, z-values drop down very fast at some points between $\mu = 3$ and $\mu = 3.2$. This means the surface stop expanding suddenly at a specific point. Though there is evidence to show that the non-existence of a turning point, this result may be caused by the large step size since the surface may still exist right before the last point of μ. Now we try to make n starts from 0 and ends at 10 with a step size of 1. Thus, we want μ to decrease with a much smaller step size from 3.1 to 3.0 in10 steps. The new result is given by the following graph

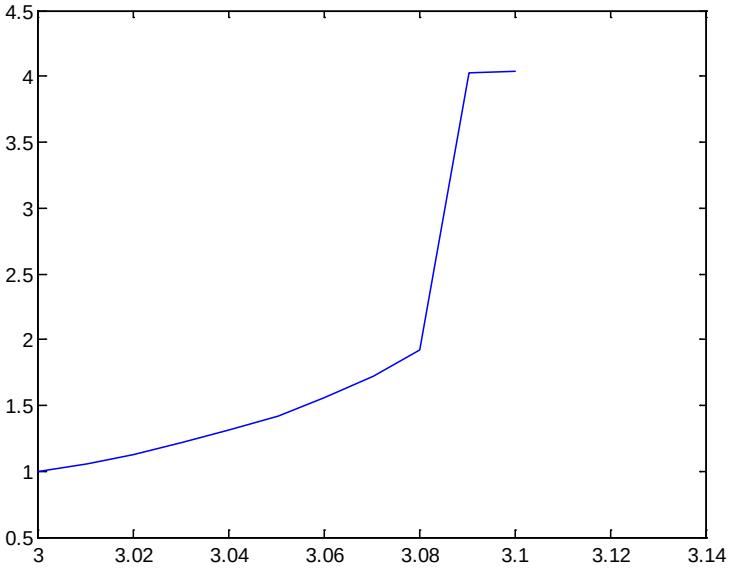

Figure 15 Decreasing μ with step size of 0.01

Unfortunately, the evidence still shows a fast dropping speed for z-values and therefore we can conclude that the surface will end suddenly at a point near $\mu = 3.09$ when μ is decreasing. Though the idea of decreasing μ fails to get an expanding surface, we can try if it will exist in the case of increasing μ. To avoid the effects of large step size, this time we try smaller step size first.

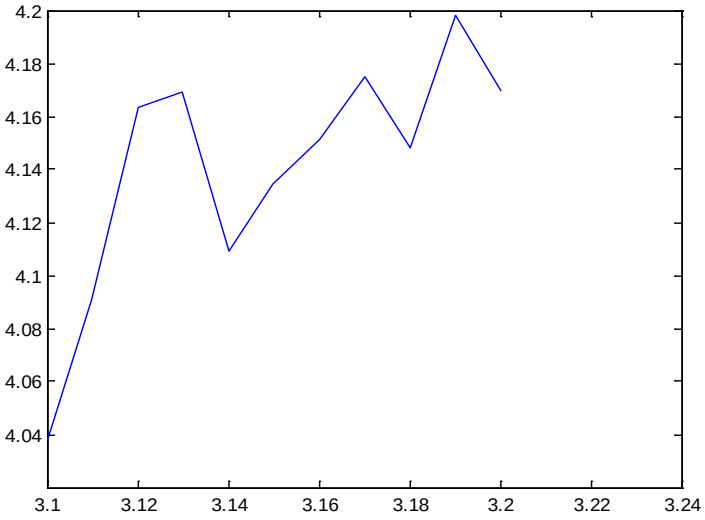

Figure 16 Increasing μ with step size of 0.01

From this graph, we successfully get a curve without fast dropping, but we still need further proof for the existence when μ continues to increase. Now, we increase the step size to be 0.1and the consequence is the following

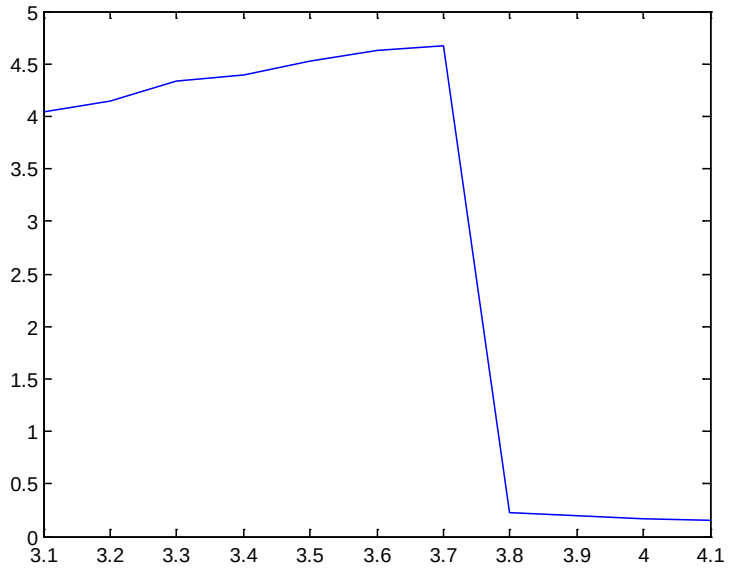

Figure 17 Increasing μ with step size of 0.1

The same as Figure 15, the fast dropping phenomenon occurs again which means when we increase the value of μ continuously, the surface will end suddenly at a point near $\mu = 3.7$.

In conclusion, the surface on the top of the 3-D graph may only appear when the value of μ is between 3.09 and 3.7.

V. Conclusion.

This research shows how the initial values affect the oscillation and bouncing behaviors when the time is very large. We use several nonlinear differential equations to model the spring system by plugging in different values to each variable, and the MATLAB outputs give us visual experience which makes the explanation easy to follow.

For the oscillations, our model is given by $y''+0.01y'+ay^{+}-by^{-}=10+\lambda\sin\mu t$. The expression of $ay^{+}-by^{-}$ adds the nonlinearity to the model cleverly. In the case of small nonlinearity and forcing, we find that when plugging in smaller initial values, a much faster shrinking speed of the amplitudes appears but the time each oscillation needs to settle down to the eventual periodic state is almost the same. When we change the nonlinearity and forcing to be large, a completely different phenomenon occurs. We can see many types of stable periodic solutions, which are depends on the initial values one has chosen. Besides these, a higher density of oscillations for each period is presented for larger initial values.

The concentration of this study is the bouncing behaviors. The main model is given by another similar nonlinear differential equation $y''+0.01y'+by=\mathrm{s}ign(y)*(a+\lambda\cos\mu t)$. In order to get the bouncing behaviors, we add an absolute value command to our original code. However, we cannot get bouncing by assigning any values to four variables. The numerical requirements to guarantee bouncing behaviors to happen are the step size between each *y* value has to be very small (less than 0.003) and the set of *y* should contain both positive and negative values. After several attempts, the ODE with $b=a=\lambda$ or $b\approx a\approx\lambda$ may often show us a good bouncing graph, but the case of $b\approx a\approx\lambda$ may include chaotic bouncing behaviors besides general bouncing. With regard to initial conditions, there is almost no effect on the bouncing. The MATLAB also shows

readers three 3-D graphs to describe the relationship for frequency, external forces and range of amplitudes. The values on z-axis stand for the range of amplitudes, and they present a larger and faster jump for each corresponding pair of λ and μ when the initial values become larger and larger. We also realize that when we increase the initial values, a possible surface appears on the top of each 3-D graph. Unfortunately, when we implement numerical techniques to analyze the slice of the surface, this statement was overthrown as we saw fast dropping z-values on both increasing and decreasing μ directions. In other words, the surface will not continuously expand as the initial values increase and it will disappear suddenly at some points.

MATLAB software contributes a lot to this study, especially for the numerical analysis. The programming code for each figure is in the Appendix.

VI. References

[1] J. Glover, A. C. Lazer and P. J. McKenna, *Existence and Stability of Large-scale Nonlinear Oscillations in Suspension Bridges*, Z.A.M.P., 40 (1989), 171-200.

[2] A. C. Lazer and P. J. McKenna, *Periodic Bouncing for a Forced Linear Spring with Obstacle*, Differential and Integral Equations, Vol. 5, No. 1 (Jan. 1992), 165-172.

[3] L. D. Humphreys and R. Shammas, *Finding Unpredictable Behavior in a Simple Ordinary Differential Equation*, The College Mathematics Journal, Vol. 31, No. 5 (Nov., 2000), 338-346

[4] Blanchard, Paul, Robert L Devaney and Glen R Hall. *Differential Equations.* Third Edition. Belmont: Thomson, 2006.

VII. Appendix

Figure 1

```
function dy=function2(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.1*y(2)-y(1)^3;

[T,X]=ode45('function2',[0 50],[20 0]);
plot(T,X(:,1));
```

Figure 2

```
function dy=function2(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.1*y(2)-y(1)^3;

[T,X]=ode45('function2',[0 200],[20 0]);
plot(T,X(:,1));
```

Figure 3

```
x=-20:0.01:0;plot(x,13*x,'-');
hold on;
x=0:0.01:20;plot(x,17*x,'-');
```

Figure 4

(1)

```
function dy=function4(t,y)

dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-15*y(1)-2*(abs(y(1)))+10+0.1*sin(4*t);

[T,X]=ode45('function4',[800 2500],[2 -1]);
plot(T,X(:,1));
```

(2)

```
function dy=function4(t,y)

dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-15*y(1)-2*(abs(y(1)))+10+0.1*sin(4*t);

[T,X]=ode45('function4',[800 2500],[1 0]);
plot(T,X(:,1));
```

(3)

```
function dy=function4(t,y)

dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-15*y(1)-2*(abs(y(1)))+10+0.1*sin(4*t);

[T,X]=ode45('function4',[800 2500],[0.1 -0.1]);
plot(T,X(:,1));
```

Figure 5

(1)

```
function dy=function4(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-9*y(1)-8*(abs(y(1)))+10+13.7*sin(0.17*t);

[T,X]=ode45('function4',[800 2000],[2 -1]);
plot(T,X(:,1));
```

(2)

```
function dy=function4(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-9*y(1)-8*(abs(y(1)))+10+13.7*sin(0.17*t);

[T,X]=ode45('function4',[800 2000],[1 0]);
plot(T,X(:,1));
```

(3)

```
function dy=function4(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-9*y(1)-8*(abs(y(1)))+10+13.7*sin(0.17*t);

[T,X]=ode45('function4',[800 2000],[0.1 -0.1]);
plot(T,X(:,1));
```

Figure 6

```
function dy=function5(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-4*y(1)+sign(y(1))*(4+4*cos(0.1*t));

timefinal=3000;
[T,Y]=ode45('function5',[0 timefinal],[2 -1],'AbsTol',eps,'NonNegative',eps);
i = min(find(T>timefinal-500));
t=T(i:end);
a=Y(i:end,1);
plot(t,a);
```

Figure 7

```
function dy=function5(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-4*y(1)+sign(y(1))*(4+4*cos(0.1*t));

timefinal=3000;
[T,Y]=ode45('function5',[0 timefinal],[2 -1],'AbsTol',eps,'NonNegative',eps);
i = min(find(T>timefinal-500));
t=T(i:end);
a=Y(i:end,1);
plot(t,abs(a));
```

Figure 8

```
function dy=function5(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-20*y(1)+sign(y(1))*(20+4*cos(0.2*t));

timefinal=3000;
[T,Y]=ode45('function5',[0 timefinal],[2 -1],'AbsTol',eps,'NonNegative',eps);
i = min(find(T>timefinal-500));
t=T(i:end);
a=Y(i:end,1);
plot(t,abs(a));
```

Figure 9

```
function dy=function5(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-10.2*y(1)+sign(y(1))*(9.9+10*cos(0.1*t));

timefinal=3000;
[T,Y]=ode45('function5',[0 timefinal],[2 -1],'AbsTol',eps,'NonNegative',eps);
i = min(find(T>timefinal-500));
t=T(i:end);
a=Y(i:end,1);
plot(t,abs(a));
```

Figure 11

(1)

```
function dy=function5(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-4*y(1)+sign(y(1))*(4+4*cos(0.1*t));

timefinal=3000;
[T,Y]=ode45('function5',[0 timefinal],[2 -1],'AbsTol',eps,'NonNegative',eps);
i = min(find(T>timefinal-500));
t=T(i:end);
a=Y(i:end,1);
plot(t,abs(a));
```

(2)

```
function dy=function5(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-4*y(1)+sign(y(1))*(4+4*cos(0.1*t));

timefinal=3000;
[T,Y]=ode45('function5',[0 timefinal],[1 0],'AbsTol',eps,'NonNegative',eps);
i = min(find(T>timefinal-500));
t=T(i:end);
a=Y(i:end,1);
plot(t,abs(a));
```

(3)

```
function dy=function5(t,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-0.01*y(2)-4*y(1)+sign(y(1))*(4+4*cos(0.1*t));
timefinal=3000;
[T,Y]=ode45('function5',[0  timefinal],[.1-.1],'AbsTol',eps,'NonNegative',eps);
i = min(find(T>timefinal-500));
t=T(i:end);
a=Y(i:end,1);
plot(t,abs(a));
```

Figure 12

(1)

```
mu=2.5:0.1:5;
lambda=0:0.05:1;
I=length(mu);
J=length(lambda);
z=zeros(I,J);
f=@(t,y,mu,lambda)[y(2);-0.01*y(2)-
10*y(1)+(sign(y(1)))*(10+(lambda)*cos((mu)*t))];
timefinal=3000;
for i=1:I
for j=1:J
[T,Y]=ode45(f,[0 timefinal],[0.1 -0.1],[],mu(i),lambda(j));
k=min(find(T>timefinal-500));
t=T(k:end);
a=Y(k:end,1);
z(i,j)=max(a)-min(a);
end
end
surf(lambda,mu,z);
zlim([0 10]);
```

(2)

```
mu=2.5:0.1:5;
lambda=0:0.05:1;
I=length(mu);
J=length(lambda);
z=zeros(I,J);
f=@(t,y,mu,lambda)[y(2);-0.01*y(2)-
10*y(1)+(sign(y(1)))*(10+(lambda)*cos((mu)*t))];
timefinal=3000;
for i=1:I
for j=1:J
[T,Y]=ode45(f,[0 timefinal],[1 0],[],mu(i),lambda(j));
k=min(find(T>timefinal-500));
t=T(k:end);
a=Y(k:end,1);
z(i,j)=max(a)-min(a);
end
end
surf(lambda,mu,z);
zlim([0 10]);
```

(3)

```
mu=2.5:0.1:5;
lambda=0:0.05:1;
I=length(mu);
J=length(lambda);
z=zeros(I,J);
f=@(t,y,mu,lambda)[y(2);-0.01*y(2)-
10*y(1)+(sign(y(1)))*(10+(lambda)*cos((mu)*t))];
timefinal=3000;
for i=1:I
for j=1:J
[T,Y]=ode45(f,[0 timefinal],[2 -1],[],mu(i),lambda(j));
k=min(find(T>timefinal-500));
t=T(k:end);
a=Y(k:end,1);
z(i,j)=max(a)-min(a);
end
end
surf(lambda,mu,z);
zlim([0 10]);
```

Figure 14

```
inia=2;
inib=-1;
for n=0:1:10
m=3.1-(n)*(0.1);
f=@(t,y,m)[y(2);-0.01*y(2)-10*y(1)+(sign(y(1)))*(10+0.5*cos((m)*t))];
timeinitial=3000*n;
timefinal=3000+timeinitial;
[T,Y]=ode45(f,[timeinitial timefinal],[inia inib],[],m);
k=min(find(T>2500+3000*n));
t=T(k:end);
a=Y(k:end,1);
z(n+1)=max(a)-min(a);
mu(n+1)=m;
inia=Y(timefinal,1);
inib=Y(timefinal,2);
end
plot(mu,z);
```

Figure 15

```
inia=2;
inib=-1;
for n=0:1:10
m=3.1-(n)*(0.01);
f=@(t,y,m)[y(2);-0.01*y(2)-10*y(1)+(sign(y(1)))*(10+0.5*cos((m)*t))];
timeinitial=3000*n;
timefinal=3000+timeinitial;
[T,Y]=ode45(f,[timeinitial timefinal],[inia inib],[],m);
k=min(find(T>2500+3000*n));
t=T(k:end);
a=Y(k:end,1);
z(n+1)=max(a)-min(a);
mu(n+1)=m;
inia=Y(timefinal,1);
inib=Y(timefinal,2);
end
plot(mu,z);
```

Figure 16

```
inia=2;
inib=-1;
for n=0:1:10
m=3.1+(n)*(0.01);
f=@(t,y,m)[y(2);-0.01*y(2)-10*y(1)+(sign(y(1)))*(10+0.5*cos((m)*t))];
timeinitial=3000*n;
timefinal=3000+timeinitial;
[T,Y]=ode45(f,[timeinitial timefinal],[inia inib],[],m);
k=min(find(T>2500+3000*n));
t=T(k:end);
a=Y(k:end,1);
z(n+1)=max(a)-min(a);
mu(n+1)=m;
inia=Y(timefinal,1);
inib=Y(timefinal,2);
end
plot(mu,z);
```

Figure 17

```
inia=2;
inib=-1;
for n=0:1:10
m=3.1+(n)*(0.1);
f=@(t,y,m)[y(2);-0.01*y(2)-10*y(1)+(sign(y(1)))*(10+0.5*cos((m)*t))];
timeinitial=3000*n;
timefinal=3000+timeinitial;
[T,Y]=ode45(f,[timeinitial timefinal],[inia inib],[],m);
k=min(find(T>2500+3000*n));
t=T(k:end);
a=Y(k:end,1);
z(n+1)=max(a)-min(a);
mu(n+1)=m;
inia=Y(timefinal,1);
inib=Y(timefinal,2);
end
plot(mu,z);
```