Pendulum Simulation 1: Period and Circular Error

Tom Van Baak, tvb@LeapSecond.com

Introduction

This is the first of several articles where I explore high-accuracy pendulum simulation.

Even now in the 21st century, a number of mysteries still exist regarding precision pendulum clocks. How do tides affect impulse? Is high-vacuum always better? How much do seismic effects contribute to clock instability? Is higher Q always better? How much do suspension losses contribute to timekeeping error? Is small amplitude always better than large amplitude? Does amplitude control compensate for rod creep? Do gravity changes conflict with amplitude control? Why does a stopped pendulum spontaneously start and how much does this spoil normal timekeeping? Would a pendulum keep better time on Mars than on Earth?

We each come to the field of precision pendulum timekeeping from different backgrounds and acquired skills. The distinct methods that we use to understand the subtle behavior of precision pendulum clocks include:

- *Historical* – Some of you are extremely well-read in horology and can solve pendulum problems or make insightful recommendations based on a deep knowledge of the past alone. There are centuries of books and articles and thousands of historical pendulum clock examples to learn from.

- *Analytical* – It is no accident that pendulums are part of every mathematics and physics curriculum. The equations are simple when simple assumptions are made; the equations get far more complex and interesting when real-world pendulum behavior is considered. I am always amazed by people who can think in equations and solve problems with only a chalk board.

- *Empirical* – The most attention goes to those who design and actually build pendulum clocks. Not just read about them, not just write equations; but get their hands dirty to bring their magnificent creations to life. Most of what we know about pendulum clocks is the result of incremental evolution. Pendulum analysis is done by physical measurements (e.g., using a Microset timer). While some read words, and others write equations, a few of you do real work!

- *Numerical* – Computers are now fast enough to model the behavior of a pendulum clock with digital simulation. The predicted behavior of a rod and bob in a gravitational field is simulated by breaking the problem into tiny segments, each representing just milliseconds or microseconds of time. The true smooth operation of a pendulum is approximated by a sequence of thousands or millions of ragged quantized steps.

Having limited skills with historical, analytical, and empirical methods I decided to see what the numerical method could offer. As a computer engineer, the *hammers* I bring to every *problem* tend to be computers, microprocessors, and software.

I figured there is no harm in seeing if numerical methods could shed light on certain pendulum behaviors. Initially I was distrustful that computer simulation of a pendulum clock could have any bearing on real world pendulum clocks. But the initial results have turned out well. For my own sake, if not yours, this article has no physics diagrams, equations, calculus, or Greek symbols. I want to describe as simply as possible the world of simulation.

Numerical simulation

The idea behind numerical simulation is incredibly simple. It is the assumption that you can break complex physical motion into tiny pieces. The bob of an idealized pendulum moves in a smooth path. In numerical simulation an imaginary bob jerks from one calculated point to another calculated point in short straight lines. Think staircase instead of smooth ramp. Worse yet, the points themselves are not precise numbers due to computer architecture limitations. How can this be good?

Computer simulation gets a poor reception in horological circles for two simple reasons: it's not real and it's not correct. It is like trying to recreate Venus de Milo out of Lego bricks: from a distance it might have the right curvy shape but up close it's a poor, ragged, embarrassing digital imitation. An analogy can be found in music (compare LP with CD or MP3), or photos (compare paintings with digital images), or video (compare live theater with DVD movies). Real is always better than digital, but over the decades the resolution of digital has improved so much that it is getting harder to tell the difference. The *retina* display is good example.

The key to understanding numerical simulation is not to worry about right and wrong, but instead to consider how close to right the wrong might be. Not every analysis of pendulum behavior needs to be done perfectly. Sometimes close enough is sufficient and so these digital, quantized approximations of motion may have some merit.

The word simulation has two meanings. You can model the parabolic path of a bouncing ball to create an animation. We see this in movies, web animations, video games, and textbooks – where a known formula or *equation of motion* is employed to predict where an object will be at any point in time. This is *not* the type of simulation we're interested in. Instead, the simulation being used here is *numerical simulation*. In this case we do not model the motion of the ball at all, but instead merely model the forces that a ball would experience. Any movement of the ball is a side effect of the simulated forces. This distinction is important.

Numerical simulation of a pendulum is very easy because there is only one force involved (gravity). Given a pendulum of constant length ***L*** and constant acceleration of gravity ***g***, just two parameters describe the bob at any given moment in time: its current angle, or amplitude ***theta*** and its current rate of motion, or velocity ***omega***.

By convention, theta is positive when right of center and negative when left of center. Similarly omega is positive when the bob is moving left to right and negative for right to left. The units for theta are radians, but often converted to degrees for us humans. The units for omega are radians per second, often converted to meters/second.

If we know the current speed and position of the bob, it is easy to predict (guess, estimate) the speed and position of the bob, say, a millisecond later with linear extrapolation. The only physics required is "rate × time = distance", which we all learned as children. The computer code I'm using for the simulation is just this:

```
repeat {
   omega -= dt * sin(theta) * g / L;    // update speed
   theta += dt * omega;                 // update angle
}
```

That's all there is to it. The constant ***dt*** is the time step size (usually a small fraction of a second). At each step, the speed of the bob is adjusted based on the force of gravity at the current angle, and angle is adjusted based on the current speed. This process repeats: speed causes a change in angle, angle causes a change in speed, speed causes a change in angle, angle causes a change in speed, etc.

We know a pendulum exhibits oscillatory motion but it is important to realize that the computer program has no idea what oscillatory motion is, what a sine wave is, what period is, or what circular error is. All the program is doing is applying simple rules of force and motion at each step. If oscillatory motion results from this, that's great (and expected). But the macroscopic behavior of the pendulum is not in any way programmed into the software. Otherwise we would be creating a pendulum animation instead of creating a pendulum simulation.

The code above implements the Euler-Cromer method. The basic Euler method of integration dates back to 1768 (nearly 250 years ago). As you might imagine, repeating imperfect calculations like this hundreds or thousands of times will certainly introduce computational errors. In 1980 Alan Cromer [1] explored subtle modifications of the Euler method and found one that was relatively immune from rounding losses! That is the version used here.

The curious reader will now search for words like: ODE, Euler integration, Euler-Cromer, Runge-Kutta, Newton–Stormer–Verlet, Computational Physics, simple pendulum, and leapfrog.

Virtual pendulum clock

Pendulum simulation software has been around as long as computers. Philip Woodward [2] wrote his own simulation programs. Bob Holmstrom [3] has experimented with commercial simulation tools. I would not doubt many others have use simulation tools as well.

My intent was not to develop a fancy, interactive, graphical, swinging pendulum program; the web is full of these. They are excellent for education but usually not good for precision pendulum research. Instead my goals were 1) to create a virtual pendulum and output raw numbers so that the data could be analyzed using the same tools and techniques that I use with raw data from real pendulum clocks, and 2) to achieve the highest possible performance and accuracy, and 3) to allow simulations as short as one swing, or as long as a day or a month in the life of a virtual pendulum clock, and 4) to allow the program to be changed to support different pendulum experiments.

The program(s) are written in C and are available on my web site [4]. They are as user friendly as any UNIX command line tool (humor). Even my 7 year old Windows XP laptop runs over 5 million steps/second. This is much faster than other pendulum simulation programs and allows me to get answers far quicker, or far more accurate, or both.

In its simplest form the program allows you to specify constants like rod length (L), the local acceleration of gravity (g), the mass of the bob (m), the initial amplitude (a), and the step size (dt). The initial value of theta is –a (pendulum pulled to the left) and the initial value of omega is 0 (pendulum is motionless). At this point the Euler-Cromer simulation loop begins, step after step after step.

At each step, the program can display the value of theta and omega. In addition, the program can report useful values such as amplitude (degrees), horizontal displacement (meters), vertical height (meters), velocity (m/s), kinetic energy (J), potential energy (J), and total energy (J). The virtual time (which is the step number times dt) is also displayed.

So now if I want to perform an experiment I have a choice of going to the basement and doing it with a real pendulum clock (Synchronome) or staying at my desk and doing it with a virtual pendulum clock. I realize that many experiments cannot be done with a virtual clock. At the same time, some experiments that might be extremely difficult or impossible to perform with a Synchronome, might be easy with a virtual clock. It is not a universal solution; just another tool.

First results

Let's see what happens if we simulate a 2-seconds pendulum having L=0.99362 m, g=9.807, m=10kg, and a=1 degree. We choose a simulation step size of 0.1 second and 20 steps are shown. The output shows step number, virtual elapsed time, bob angle (degrees), height (meters), velocity (meters/second), and total energy (Joules):

```
step    seconds      angle     height   velocity     energy
   0   0.000000  -1.000000   0.000151   0.000000   0.014841
   1   0.100000  -0.901305   0.000123   0.017116   0.013521
   2   0.200000  -0.713656   0.000077   0.032542   0.012854
   3   0.300000  -0.455570   0.000031   0.044757   0.013096
   4   0.400000  -0.152521   0.000004   0.052555   0.014155
   5   0.500000   0.165583   0.000004   0.055165   0.015623
   6   0.600000   0.467343   0.000033   0.052331   0.016934
   7   0.700000   0.722977   0.000079   0.044332   0.017584
   8   0.800000   0.907256   0.000125   0.031957   0.017322
   9   0.900000   1.001992   0.000152   0.016429   0.016250
  10   1.000000   0.997837   0.000151  -0.000721   0.014780
  11   1.100000   0.895201   0.000121  -0.017799   0.013478
  12   1.200000   0.704212   0.000075  -0.033121   0.012845
  13   1.300000   0.443720   0.000030  -0.045174   0.013126
  14   1.400000   0.139433   0.000003  -0.052769   0.014212
  15   1.500000  -0.178616   0.000005  -0.055156   0.015684
  16   1.600000  -0.479036   0.000035  -0.052099   0.016977
  17   1.700000  -0.732175   0.000081  -0.043899   0.017592
  18   1.800000  -0.913051   0.000126  -0.031367   0.017292
  19   1.900000  -1.003813   0.000152  -0.015740   0.016193
  20   2.000000  -0.995504   0.000150   0.001441   0.014718
```

We can immediately make the following observations:

- The virtual pendulum is swinging! See how it starts at –1 degree, swings right through 0, hits +1 degree, and heads back left through 0, and ends up near –1 degree. Plotting the data would give a cosine curve.
- The velocity starts small, increases to about +55 cm/s near center, and then decreases to near zero at the end of the swing. It then increases to about –55 cm/s near the center as it

goes left and ends up near zero at the end of the period. Plotting the data would give a sine curve.

- The total energy (PE + KE) is roughly 15 mJ, but clearly is not constant.
- The period is about 2 seconds, as expected for this pendulum.

The bad news is that while the amplitude starts at exactly –1 degree and doesn't end at exactly +1 degree. Worse yet, total energy is not very constant. Errors in precision like this are one example why simulation is often not treated seriously. However, these poor results are to be expected with a step size of only 0.1 second. If instead we reduce the step size to 0.0001 second (0.1 millisecond), the program computes 20,000 points. Below only every 1000th step is shown:

```
 step    seconds      angle     height    velocity     energy
    0   0.000000  -1.000000   0.000151   0.000000   0.014841
 1000   0.100000  -0.951009   0.000137   0.016835   0.014840
 2000   0.200000  -0.808927   0.000099   0.032023   0.014839
 3000   0.300000  -0.587661   0.000052   0.044076   0.014839
 4000   0.400000  -0.308870   0.000014   0.051815   0.014840
 5000   0.500000   0.000156   0.000000   0.054482   0.014841
 6000   0.600000   0.309167   0.000014   0.051815   0.014843
 7000   0.700000   0.587914   0.000052   0.044076   0.014843
 8000   0.800000   0.809110   0.000099   0.032023   0.014843
 9000   0.900000   0.951105   0.000137   0.016836   0.014843
10000   1.000000   1.000000   0.000151   0.000000   0.014841
11000   1.100000   0.951009   0.000137  -0.016835   0.014840
12000   1.200000   0.808928   0.000099  -0.032023   0.014839
13000   1.300000   0.587662   0.000052  -0.044076   0.014839
14000   1.400000   0.308871   0.000014  -0.051815   0.014840
15000   1.500000  -0.000155   0.000000  -0.054482   0.014841
16000   1.600000  -0.309166   0.000014  -0.051815   0.014843
17000   1.700000  -0.587913   0.000052  -0.044076   0.014843
18000   1.800000  -0.809109   0.000099  -0.032023   0.014843
19000   1.900000  -0.951105   0.000137  -0.016836   0.014843
20000   2.000000  -1.000000   0.000151  -0.000000   0.014841
```

Now we see wonderful results. The angle starts at –1 degree, goes to +1 degree, and then back to –1 degree, exactly. The sum of PE and KE at every step is now 14.844 mJ, consistent to 5 or 6 decimal places. So as long as we use a small enough step size the virtual pendulum starts acting like a real pendulum.

Accuracy and patience

Fortunately, using a smaller step size is no problem at all. The program is so efficient it can compute a million steps in a fraction of a second; a billion steps take only a few minutes. Thus the tool can be used to perform incredibly precise or extremely long simulations.

So is simulation valid? Is approximation tolerable? Even with a real pendulum there are implicit errors in measurement. Nothing is perfect. After you play with a virtual pendulum you realize it can be more accurate than a real pendulum in many cases.

As an example, if I wanted to investigate the effect of tides I could spend years trying to make a world-class pendulum that would be so accurate (about 8 decimal places) that it would detect the variations in gravity due to the motions of the moon-earth-sun system. Or I could pick a small enough step size so my virtual pendulum clock was accurate to 8 decimal places and then just vary gravity in software. You can see where I'm heading.

The choice of accuracy is dictated by what experiment you're trying to do. This is true for both real and virtual pendulum clocks. The difference is that greater accuracy is easy to achieve in software pendulums and much harder to achieve with hardware pendulums. In most cases one simply trades patience for accuracy. For extreme accuracy the computer can be left to run for many hours instead of a few minutes or seconds.

Virtual pendulum measurement

With a real pendulum we usually make measurements of beat or period, of peak amplitude or peak velocity. The measurements can be made as often as once a second although usually they are averaged and reported as less frequent intervals for convenience.

Virtual pendulum software can report the position of the virtual bob each step. But that is usually far too much data. Instead it is desirable to extract only the peak amplitude, peak velocity, and period of a virtual pendulum. How can this be done? At every step we know theta and omega. Peak amplitude occurs at the end of the swing. With simple comparison, a computer program can determine which point is the peak and report it.

Another technique is even easier. The center point of the swing occurs when theta changes sign. Similarly, the end points of the swing occur when omega changes sign. After each step, a check for sign change in omega or theta is made to determine if the virtual pendulum is at one of these measurement points. These measurements, as well as the step count, can be made more precise with linear interpolation over the zero-crossings.

It is as if the virtual pendulum has a built-in virtual Microset timer that automatically reports period, peak amplitude and velocity every swing. In this way raw data from a virtual pendulum looks the same as raw data from a real pendulum; they can both be processed by the same graphing, analysis, and statistical tools.

Period and circular error

As an example of the built-in measurement capability, let's measure the period of a virtual pendulum (L=1 m, g=9.8 m/s^2, a=1). The expected answer is 2.007128135980991 seconds.

step size = 1e-2 s, measured period = 2.00704 (error = 4.1e-5)
step size = 1e-4 s, measured period = 2.007128127 (error = 4.1e-9)
step size = 1e-6 s, measured period = 2.00712813598018 (error = 4e-13)

You can see that amazing accuracy is possible with even modest step sizes. 12 digit accuracy was obtained with 2 million steps in less than half a second.

Remember that the Euler-Cromer method knows absolutely nothing about period or circular error; it is simply following basic rules of straight line motion in small steps. So does a virtual pendulum exhibit circular error like a real pendulum does? To test this we can try different initial amplitudes and see how the period compares with the expected known value.

Here is the result for amplitude 1' (one minute of angle):
    2.007089933768982 – using pendulum period formula
    2.00708993376893 8 – measured with virtual pendulum

Here is the result for amplitude 1° (one degree of angle):
  2.007128135980991 – using pendulum period formula
  2.00712813598085 – measured with virtual pendulum

Here is the result for amplitude 15°:
  2.015721572277283 – using pendulum period formula
  2.01572157227766 – measured with virtual pendulum

The accuracy is 13 or 14 digits; a virtual pendulum exhibits circular error as it should.

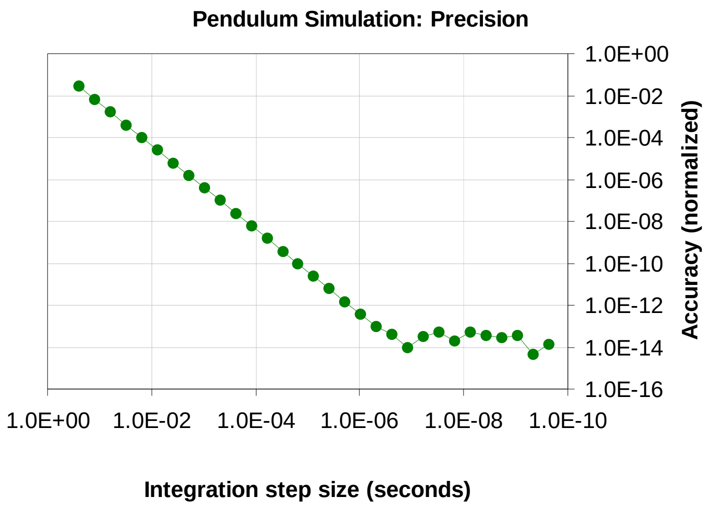
At extremely large angles, high precision is harder to obtain. Here is amplitude 179°:
  7.829788572986007 – using pendulum period formula
  7.829788573177146 – measured with virtual pendulum

In this case the accuracy is "only" about 11 digits after a few minutes of computation. No one builds a real pendulum clock with 179° amplitude but this works fine with simulation. So far, the virtual pendulum has matched the predictions of the AGM-pendulum formula for every possible combination of L or g or angle I can think of.

Limits to accuracy

We have seen that simulation accuracy depends on step size. This does not mean, however, that one can achieve infinitely good accuracy by using ever smaller and smaller steps. There has to be a limit. We know this because of the way native computer arithmetic works. In this case I'm using double precision, floating-point math, which has 53-bits of binary precision (about 15.9 decimal digits). A simple experiment was performed in order to verify that these limitations exist, and to measure them.

An ideal precision pendulum with L=1m, g=9.8m/s², a=1° should have period of 2.007128135980991 seconds. The pendulum simulator was run 30 times, with a different step size each time. The step size ranged from as large and unrealistically coarse as 2^{-1} (½ second) to as small and impractically fine as 2^{-30} (about 1 nanosecond). The graph shows how the virtual period compares to the pendulum period equation for all 30 runs:



The results are wonderful. First, it clearly shows that accuracy improves as step size decreases. Second, it shows that accuracy improves at double the rate of step size: a 1 millisecond step size gives period with 6 digits of accuracy, and a 1 microsecond step size gives period with 12 digits of accuracy. Finally, the graph shows there is a 13 or 14 digit limit of accuracy, regardless of step size. All this is expected, which gives further confidence to this numerical method.

A precision "floor" of 14 digits poses no problem for my intended use of the program. Note this is far more precise than any real pendulum or electronic pendulum measurement system.

Conclusion

So it is possible to create a virtual precision pendulum clock through numerical simulation. The virtual pendulum matches standard analytical formulas and empirical experiments to a high degree. This idealized (constant gravity, friction-less, etc.) virtual pendulum:

- swings back and forth, as expected
- has period which matches the pendulum formula
- maintains constant amplitude, as expected
- exhibits circular error, exactly as predicted
- conserves total energy, as expected
- works for any angle of amplitude, small or large
- becomes more accurate as step size decreases
- works with step sizes as low as 100 nanoseconds
- produces answers up to 14 digits of accuracy
- can simulate just one swing or many days or months

I am pleased with the results. I did not realize numerical simulation was so easy or could be made so accurate and efficient. Having gained confidence in this method, the plan is to apply this to some of the pendulum mysteries I am interested in. I realize that some of these questions could be answered with advanced mathematics or by experimenting with real pendulum clocks, but I think the virtual pendulum clock is not without promise.

Simulation has many limitations. I don’t think it can be used to compare spring or knife-edge suspension, or figure out the best bob shape, etc. Instead I plan to use the tool only for the class of problems where it might help; problems that involve forces, geometry, energy, and noise. These will be described in subsequent articles [5] in this series.

Notes:

[1] http://en.wikipedia.org/wiki/Euler-Cromer_algorithm

[2] Philip Woodward, *Experiments with a simulated clock*, Horological Journal, January 1991, pages 240-242

[3] Bob Holmstrom, *Useful Tool for Horological Modeling*, HSN 1999-1, pages 18-21

[4] www.leapsecond.com/tools/pend8.exe (pend8.c) is a simple simulator

[5] Copies of my HSN pendulum papers: http://leapsecond.com/hsn2006/