Formal Methods in the Verification and Validation of Simulation Models

K. E. Kennedy*
Department of Computer Science
Clemson University
Clemson, SC 29634-1906
kkennedy@cs.clemson.edu

Keywords
verification, validation, simple pendulum, simulation models

ABSTRACT

We examine the use of formal methods in the verification and validation of simulation models. The verification methods covered are deductive inference and model checking. The validation methods are statistical tests and inductive inference.

The model that the methods are demonstrated on is the simple pendulum problem. This problem was chosen because it is conceptually easy to understand and allows us to focus on the verification and validation methods.

1. INTRODUCTION

In this paper, we examine formal methods to verify and validate a simulation model. To illustrate the use of the formal methods that will be covered, we will study the classical simple pendulum problem and how each method can be used to verify or validate this model.

The purpose of verification is often stated as "are we building the model right?" while validation is considered as "are we building the right model?"

The purpose of using formal methods in verification and validation is to make the process of verifying and validating the simulation model a quantitative, computational process. Too often, verification and validation are only qualitative. While it is often fine to use qualitative measures for small simulation models, qualitatively verifying and validating a simulation model of substantial complexity can become impractical because of all of the variables present in the model.

A review of the simple pendulum problem is included in Section 2. Verification of the pendulum problem is explained in Section 3, and validation of the model is covered in Section 4. We present our conclusions about formal methods in verification and validation of the simple pendulum problem, and of simulation models in general, in Section 5.

2. SIMPLE PENDULUM PROBLEM

*Partial funding for this work was provided by NSF Grant DUE-0127488, the Shodor Education Foundation, Durham, NC and the Shodor Computational Science and Engineering Fund at Clemson University, Clemson, SC.

The pendulum problem has often been discussed in modeling. A graphical representation of a pendulum is given in Figure 1 where θ is the angle of displacement, s is the amount of displacement on the arc, L is the length of the string, m is the mass of the object attached to the string, $\mathbf{T}$ is the tensional force on the string, and $\mathbf{g}$ is the force of gravity.

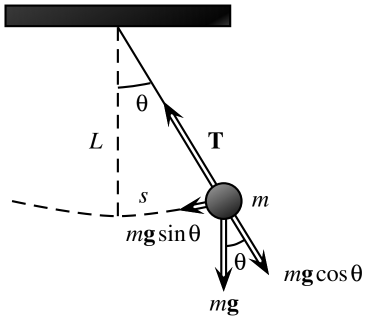

Figure 1: The simple pendulum.

There are two simple pendulum models. These are the simple pendulum with arbitrary angle assumption which we discuss in 2.1 and the simple pendulum with small angle assumption which is explained in 2.2. [13] contains more information on other pendulum models.

2.1 Arbitrary Angle Assumption

The simple pendulum with arbitrary angle assumption is the model derived from Figure 1. As one can see from the figure, the tangential force is

$$F_t = -m\mathbf{g}\sin\theta \tag{1}$$

The oscillation period is

$$T = 4\sqrt{\frac{L}{\mathbf{g}}}\int_0^{\frac{\pi}{2}} \frac{1}{\sqrt{1-\sin^2\frac{\theta_0}{2} * \sin^2\psi}}\,d\psi \tag{2}$$

which can be calculated by the infinite series

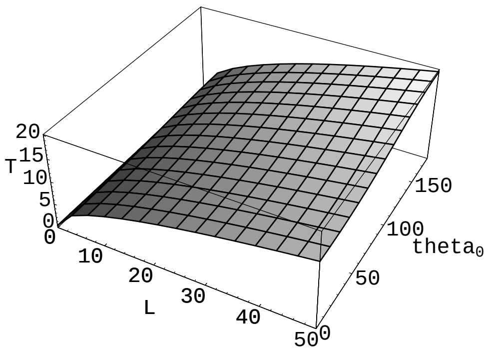

Figure 2: Three dimensional graph of the oscillation period for the simple pendulum with arbitrary angle assumption.

$$T = 2\pi\sqrt{\frac{L}{\mathbf{g}}}\left(1 + \frac{1}{4}\sin^2\frac{\theta_0}{2} + \frac{9}{64}\sin^4\frac{\theta_0}{2} + \ldots\right) \qquad (3)$$

where θ_0 is defined as the maximum angular displacement in radians. For the simple pendulum with arbitrary angle assumption, θ_0 can be any value.

A graph of the oscillation period for which it is assumed $\mathbf{g} = 9.81$ is shown in Figure 2.

2.2 Small Angle Assumption

The simple pendulum with small angle assumption is a restricted model of the simple pendulum with arbitrary angle assumption described in Section 2.1. When the angle of displacement is small, say $\theta \leq 10^\circ$, then $\sin\theta \approx \theta$, and Equation 1, for the tangential force, can be rewritten as

$$F_t = -m\mathbf{g}\theta \qquad (4)$$

The simplified tangential force equation can be transformed to show that it is simple harmonic motion. When using Equation 4 for the simplified tangential force equation, Equation 2, for the oscillation period, can be written as

$$T = 2\pi\sqrt{\frac{L}{\mathbf{g}}} \qquad (5)$$

The oscillation period for the simplified tangential force depends only on the force of gravity and the length of the pendulum's string. So unlike the oscillation period for the simple pendulum with arbitrary angle assumption, the oscillation period is easy to calculate numerically.

A graph of the oscillation period, where $\mathbf{g} = 9.81$, is shown in Figure 3.

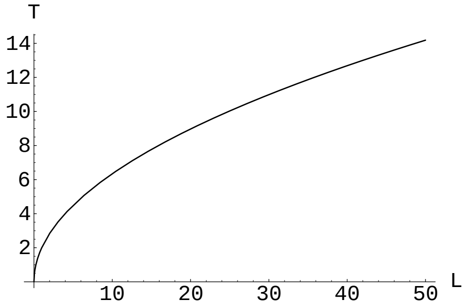

Figure 3: Two dimensional graph of the oscillation period for the simple pendulum with small angle assumption.

The reason that the simple pendulum with small angle assumption is valid can be seen by examining the curve in Figure 3 and comparing it to the surface in Figure 2. As θ_0 increases in Figure 2, the period increases. So for $\theta > 10^\circ$, the simple pendulum with small angle assumption underestimates what the period should be.

3. VERIFICATION METHODS

The purpose of formal verification is to prove the correctness of the simulation model by producing a proof from the simulation model's set of specifications. It is important to realize that a proof of correctness of the code does not validate the model. All the proof will guarantee is that for the given specifications, the code is correct.

For formal verification, the program will need to be in a language that can be formally verified. The common languages for this are first-order predicate calculus (FOPC) and temporal logic. For the simple pendulum model, we chose to write it in temporal logic.

There are modeling languages such as DEVS [15] in which work has been done to informally verify the specifications [14]. The disadvantage of informal verification is that it will only guarantee the program for the conditions that it has checked. It will not guarantee the correctness of the program for all situations.

Another issue with formal verification is that no observational data is required. We do not care how well the simulation performs for verification, we are only interested in the logical correctness of the specifications.

The two fundamental methods used to formally verify a program are deductive inference and model checking. We shall look at how deductive inference can be used in verifying a simulation model in Section 3.1 and then see how model checkers can verify a simulation model in Section 3.2.

3.1 Deductive Inference

Deductive inference, also known as theorem proving, is where we have $P \vdash C$. We are given, or can derive, the premises P, and we want to get the logically derived conclusion C.

An argument is valid if the conclusion will be true when the premises are true. An argument is sound when its premises are true and it is valid. Deductive inference is sound, so given a set of true premises and a valid argument, we are always guaranteed to get a true conclusion.

There are many inference rules in deductive inference. The reader can consult any general logic text such as [6] for a review of the inference rules.

Most theorem provers are designed to work on expressions in the form of FOPC. The two dominant theorem proving methods are resolution [12] and tableaux [3].

Today, there are two approaches to theorem proving. The first approach is to automate the entire theorem proving process. A theorem prover that demonstrates this is Otter [5]. The second approach is to use human intervention with semi-automated theorem provers. The latter method has become popular because automated theorem proving is possible but can be costly in terms of time when the theorem prover is given a long set of specifications. Examples of the second approach are Isabelle and HOL [11].

To verify the simulation model, we must either write the model in FOPC, or we can write it in a programming language such as C and then convert the code to FOPC.

Since the late 1960's, methods have existed for converting FOPC statements into executable code [4]. The advantage of this method is that the simulation model can be dealt with in more abstract terms. The disadvantage is that the code is often not as efficient as it would be if a programmer had written it.

3.2 Model Checking

Model checking is an automated method to check finite state systems. A review of model checking is given in [2]. Model checking has been used at this point mostly in the verification of circuits, but it can be used in software verification as well. A model checker searches for counter-examples. If no counter-example is found, then the model being verified is said to be correct according to the specifications.

The language of the specifications for model checkers is usually some form of temporal logic. The specifications will need to include many attributes about the model such as assumptions and fairness conditions.

Model checkers are usually entirely automated. We saw in Section 3.1 that automated theorem provers are often inefficient. There is a similar problem with model checkers which is known as the state space explosion where there is a large interaction of many different components.

Modern model checkers, such as SMV [8], have tried to get around the state space explosion problem by using Bryant's ordered binary decision diagrams (OBDDs) [1], but it still remains an issue, so anyone using model checkers must be aware of the problem and realize that to verify some systems may require a lot of time and memory.

As mentioned above, the simple pendulum with small angle assumption model was written in temporal logic. The model checker used for verification was SMV. Due to the simplicity of the model, the SMV system did not take long to verify the specifications.

4. VALIDATION METHODS

The validation of a simulation model must necessarily deal with uncertainty. Due to this, validation has historically been a much more difficult problem to quantify than verification. With verification, we can be certain if the model is correct or not according to the given specifications. With validation, the best we can do is give confidence limits for the simulation model.

Run	**1**	**..**	**10**
String Length (m)	0.734	..	0.469
Mass (kg)	0.100	..	0.050
Amplitude	0.101570	..	0.058147
Period	1.7366	..	1.3746
Release Angle	10	..	5

Table 1: Table of tests on the simple pendulum. Data provided by Dr. Jośe D'Aruda at UNC Pembroke.

A significant issue with validation is that observational data is required. For some models, this can add a significant cost to the validation process. For other models, it may not be possible to obtain the necessary data. An example is modeling how the nuclear stockpiles of the United States will age. Only limited observations are available because this is the first time that we have had to deal with aging nuclear weapons, so validating the model becomes a significant problem.

In this section, we will look at validating the model using statistical tests in Section 4.1 and inductive inference in Section 4.2.

4.1 Statistical Tests

A natural way to deal with uncertainty is through statistics. A review of statistics can be found in [10]. Statistical methods have been applied successfully to several areas where uncertainty is involved. One such example is natural language understanding [7]. Consequently, to deal with the uncertainty present in simulation models, many researchers have begun using statistics for validation. In Table 1, we have a sample of data collected in controlled experiments on a pendulum.

With the data, we will perform a linear regression where we assume the model is Equation 5. We take the period to be the response variable and the square root of the string length to be the predictor variable. The constant of the predictor variable should be $\frac{2\pi}{\sqrt{g}}$.

A scatter plot with the regression line is shown in Figure 4.

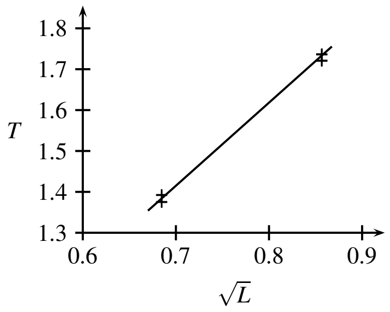

Figure 4: Regression of Period vs. Square Root of String Length.

From the regression analysis, we have the fitted model

$$\hat{Y} = 0.00892 + 2.00754X \quad (6)$$

The r^2 value of the regression is 0.9978 showing that almost all of the variation in Y can be explained by the X variable, and the MSE, the estimated variance, is $7.88e^{-5}$.

There is an argument for using the no-intercept model. When the string length is 0, we would expect the period to be zero as well. But as one can see from the fitted model, the b_0 parameter is approximately 0, and a formal test shows that the b_0 parameter is not significantly different than 0. Therefore, it was unnecessary to use the no-intercept model.

The b_1 parameter may be interpreted as for every unit increase in the square root of the length of the pendulum string, the period will increase by 2.00754.

To validate the model, we will need to see if the b_1 parameter is the same as $\frac{2\pi}{\sqrt{\mathbf{g}}}$. In order to test this, we will use the t test where we have the hypotheses

$$H_0 : \beta_1 = \beta_{10} \quad (7)$$
$$H_a : \beta_1 \neq \beta_{10} \quad (8)$$

in which β_1 is estimated by b_1. The test statistic we will use is

$$t^* : \frac{b_1 - \beta_{10}}{s_{b_1}} \quad (9)$$

We will conclude H_0 if $t^* \leq t_{(1-\frac{\alpha}{2};n-2)}$ and H_a otherwise.

We know from the regression $b_1 = 2.00754$, $s_{b_1} = 0.03334$, and, from Equation 5, $\beta_{10} = \frac{2\pi}{\sqrt{g}}$. The significance level, α, will be set at 0.01. So assuming $\mathbf{g} = 9.81$, we have

$$t^* : \frac{2.00754 - 2.00607}{0.03334} = 0.04419 \quad (10)$$

and the critical value is

$$t_{(1-\frac{\alpha}{2};n-2)} = 3.355 \quad (11)$$

resulting in

$$t^* < t_{(1-\frac{\alpha}{2};n-2)} \quad (12)$$

so we conclude H_0 from Equation 7. In other words, with 99% confidence, we are certain that $\beta_1 = \frac{2\pi}{\sqrt{g}}$ and that our model is correct.

4.2 Inductive Inference

There are two ways that one can view inductive inference. The first is when we have a model, and we wish to use the model to predict new observations. The second is when we have observations, and from the observations, we derive a model.

The second method is known as machine learning. A good overview of several popular machine learning algorithms and how to evaluate them is in [9]. Machine learning algorithms are useful when one has a lot of data but does not know what the model is. This section focuses on prediction intervals for inductive inference.

The calculation of prediction intervals is another standard statistical method. There are two ways in which prediction intervals are normally calculated. The first is to consider one new observation. The second is to consider m new observations. In this section, we consider the prediction interval for only one new observation which is calculated by

$$\hat{Y}_h \pm t_{(1-\frac{\alpha}{2};n-2)} s\{pred\} \quad (13)$$

in which $s^2\{pred\}$ is defined as

$$s^2\{pred\} = MSE\left[1 + \frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}\right] \quad (14)$$

In Equation 14, n is the number of observations used in the regression analysis, X_h is the X value of the new observation to be predicted, and $\bar{X}$ is the mean of the X values of the observations used in the regression analysis.

If we want to calculate prediction intervals for when the string length is 0.650, and for which the square root is 0.806, then we would have

$$s^2\{pred\} = 7.88e^{-5}\left[1 + \frac{1}{10} + \frac{(0.806 - 0.788)^2}{0.071}\right] \quad (15)$$

in which $s^2\{pred\} = 8.70e^{-5}$. At $\alpha = 0.01$ we get the prediction interval

$$1.60 \leq \hat{Y}_h \leq 1.66 \quad (16)$$

Thus, we would expect 99% of the new observations for when the string length is 0.650 to fall within this range. If this was not observed after a large number of new observations, then we would need to consider that the model is not valid.

Prediction intervals can also be calculated for the entire graph as shown in Figure 5.

If one is unable to get new observations but still wants to use inductive inference to see how well the model predicts, then the original data set can be divided into a training and testing set – provided that the original set is large enough. The training set is used to create the model, and the testing set is used to see how well the model predicts.

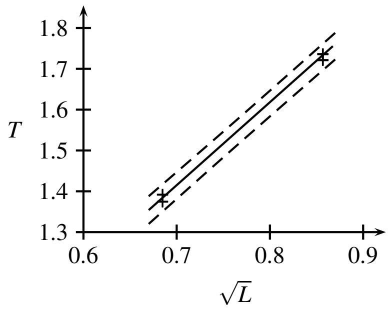

Figure 5: Prediction Intervals for the Regression of Period vs. Square Root of String Length with $\alpha = 0.01$.

5. CONCLUSIONS

We have shown only a few of the possible methods for verification and validation. There are many other tests that one might want to run to make sure that a model is valid. These tests include looking at the distribution of the data and seeing if the residuals exhibit multicollinearity or non-constant error variance.

This paper focused on the verification and validation of the simple pendulum with small angle assumption. The simple pendulum with arbitrary angle assumption would be similar. While we can not calculate Equation 3 for all terms, we can calculate it for the first k terms where k is dependent upon the accuracy of the results.

As the complexities of simulations increase, and we rely more upon the results that we get from the simulation models, formal methods will be necessary to ensure that we can believe the results of the models. However, the amount of time required to formally verify and validate simulation models also increase as the models become more complex.

6. REFERENCES

[1] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[2] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[3] M. C. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, New York, 2nd edition, 1996.

[4] C. Green. Theorem-proving by resolution as a basis for question-answering systems. *Machine Intelligence*, 4:183–205, 1969.

[5] J. A. Kalman. *Automated Reasoning with Otter*. Rinton Press, 2001.

[6] C. G. Luckhardt and W. Bechtel. *How to do Things with Logic*. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1994.

[7] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[8] K. L. McMillan. *The SMV System*. PhD thesis, Department of Computer Science, Carnegie-Mellon University, 1992.

[9] T. M. Mitchell. *Machine Learning*. WCB McGraw Hill, Boston, Massachusetts, 1997.

[10] J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman. *Applied Linear Statistical Models*. McGraw-Hill, 4th edition, 1996.

[11] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[12] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[13] R. A. Serway. *Physics for Scientists and Engineers*. Saunders College Publishing, Philadelphia, 4th edition, 1996.

[14] G. Wainer, L. Morihama, and V. Passuello. Automatic verification of devs models. In *Proceedings of the 2002 Spring Simulation Interoperability Workshop*, Orlando, FL, 2002. SISO Publisher.

[15] B. Zeigler, Y. Moon, D. Kim, and G. J. Ball. The devs environment for high performance modeling and simulation. *IEEE Computational Science and Engineering*, 4(3), July–September 1997.