AN EVALUATION SCHEME FOR THE UNCERTAINTY ANALYSIS OF A CAPTIVE TRAJECTORY SYSTEM

Sean Tuling
Defencetek, CSIR

Keywords: *Uncertainty Analysis, CTS*

Abstract

A generalised uncertainty analysis scheme for the captive trajectory system of the Medium Speed Wind Tunnel of the CSIR has been devised. The scheme uses the trajectory generation data reduction code to determine the uncertainty of a parameter. The scheme is implemented using Matlab, utilising the symbolic toolbox, the Maple kernel and object oriented concepts. The scheme represents the parameters as symbolic objects, allowing the scheme to determine the equations of derived parameters under consideration that are only dependent on system independent parameters from the trajectory generation data reduction code. A test case using a trajectory generated by the 1/15th scale stable store being released from the NACA wing-body model was performed to assess the implementation.

List of Symbols

b	Bias error variance
B	Bias limit
C_a	Axial force coefficient
C_l	Moment coefficient about the store x body axis
dz	Vertical displacement between store and pylon
K	Coverage factor
l_{ref}	Reference length
M	Mach number
M_x	Moment about the store x body axis
P	Precision limit
P_s	Tunnel static pressure
P_t	Tunnel total pressure
q	Dynamic pressure
S	Precision error variance, reference area
T_t	Tunnel total pressure
U	Uncertainty
W_i	Store body axis velocity in the z axis direction
X_I	Translational displacement of the store centre of gravity in the inertial x axis direction
Y_I	Translational displacement of the store centre of gravity in the inertial y axis direction
Z_I	Translational displacement of the store centre of gravity in the inertial z axis direction
δ_{ik}	Kronecker delta
ΔM	Tunnel mach number correction
ψ_I	Rotational displacement of the store about the inertial z axis
ϕ_I	Rotational displacement of the store about the inertial x axis
ρ	Correlation coefficient
θ_I	Rotational displacement of the store about the inertial y axis

Nomenclature

CTS	Captive trajectory system
HAOA	High angle of attack
MMS	Main model support
MSWT	Medium Speed Wind Tunnel
PC	Personal computer
RAM	Random access memory
SWS	Side wall support

1 Introduction

In experimental testing, the presentation of results requires the qualification of the results with uncertainty bounds, error bars or an error bound. This is because the true value of a result can never be determined through experimental testing. Only an estimate of the true value is obtained, with the measured valued being offset from the true value. Uncertainty bounds provide the reader of the results with an estimate that the true value lies, within a given confidence level, somewhere between the two extremes of the bounds defined.

The qualification of experimental results with uncertainty bounds provides the reader information whereby the data can be interpreted more meaningfully than without it. Indeed without uncertainty bounds, experimental results are meaningless because the results cannot be placed within the context of the experimental test itself let alone applying the results to other applications. Uncertainty bounds allow results from a repeated experiment to quantitatively be assessed for repeatability.

Within the context of captive trajectory testing, uncertainty bounds qualify the meaning of trajectories by indicating to the reader whether changes in a displacement state variable are significant or not. Alternatively put, uncertainty bounds can help the reader not to place effort in trying to interpret trends which may not necessarily be significant.

While the primary motivation for providing uncertainty bounds with trajectories in captive trajectory testing is to qualify the results obtained, the uncertainty analysis process can provide insight into the parameters that contribute to the uncertainty of a state variable. The uncertainty analysis process can thus move from being reactive to proactive i.e. from qualifying results to identifying the major contributing elements in the uncertainty of a result to designing experiments that can meet expected requirements.

With the emphasis by industry to present results with uncertainty bounds, the CSIR has implemented an initial attempt at providing captive trajectory test results with uncertainty bounds. The implementation was chosen to be flexible so as to accommodate code changes without having to rederive the required formulae. This necessitated that a generalised uncertainty analysis scheme be devised. While a generalised uncertainty analysis scheme can be devised, the efficient and effective implementation of the scheme is facilitated using modern computing developments.

This paper covers the principles implemented in the generalised uncertainty analysis scheme. Details of the implementation are, however, also elucidated namely the use of symbolic mathematics and object oriented concepts.

2 Background

The Medium Speed Wind Tunnel (MSWT) at the CSIR is a closed–circuit variable density transonic facility, with a Mach number range of M=0.3 to M=1.4. The total pressure range is 20kPa to 250kPa. The MSWT has a 1.5m x 1.5m slotted test section with a 5% porosity. The test section is equipped with four model support systems namely the main model support (MMS), captive trajectory system (CTS), side wall support (SWS) and a high angle of attack (HAOA) rig. The CTS is used to mount the store models for the captive trajectory system while the SWS is used for half model testing. The MMS is the predominant test article support system, having a pitch range of -10^o to +30^o and a roll range of -180^o to +180^o. The HAOA rig extends the pitch range of the MMS by adding a extra degree of freedom resulting in a pitch range of -7^o to +60^o. The CTS has three independent translational and three independent rotational degrees of freedom. The ranges of the various degrees of freedom are listed in Table 1.

Table 1 CTS Operational Range

Degree of freedom	Range	Units
Pitch	±45	degrees
Yaw	±45	degrees
Roll	±180	degrees
X	±560	mm
Y	±410	mm
Z	±525	mm

2.1 Theoretical Background

The theoretical background presented has been obtained from references [1], [2] and [3]. While the theoretical development given in these references is more complete, only the relevant equations are shown here.

An error (the difference between the experimentally determined value and the true value) consists of two components, namely a bias or systematic component, B_i and a precision or random component, P_i [1].

The uncertainty, U_i, of an experimental value about the measured value of X_i is given by:

$$U_i = \sqrt{B_i^2 + P_i^2} \tag{1}$$

The precision limit, P_i, for measured variable, X_i, is given by:

$$P_i = KS_i \tag{2}$$

where S_i is the standard deviation and K is the coverage factor and equals 2 for 95% confidence level [1]. The use of $K = 2$ assumes a large sample size and Gaussian error distribution.

The uncertainty in an experimental result, r, (whose result is obtained through a series of data reduction equations) is a function of j variables X_i, obtained from [1] and [2] and is represented as:

$$r = r(X_1, X_2, ..., X_j) \tag{3}$$

The uncertainty in the result, r, is, thus, as stated in [2]:

$$U_r = \sqrt{(\frac{\partial r}{\partial X_1}U_{x1})^2 + ... + (\frac{\partial r}{\partial X_j}U_{xj})^2} \tag{4}$$

where U_{xi} are the uncertainties in the measured variables X_i.

The generalised combined standard uncertainty equation is defined as:

$$\begin{aligned} u_c^2 = & \sum_{i=1}^{j}\left(\left(\frac{\partial r}{\partial X_i}b_i\right)^2 + \right. \\ & \left. \sum_{k=1}^{j}\frac{\partial r}{\partial X_i}\frac{\partial r}{\partial X_k}\rho_{b_{ik}}b_i b_k(1-\delta_{ik})\right) + \\ & \sum_{i=1}^{j}\left(\left(\frac{\partial r}{\partial X_i}S_i\right)^2 + \right. \\ & \left. \sum_{k=1}^{j}\frac{\partial r}{\partial X_i}\frac{\partial r}{\partial X_k}\rho_{S_{ik}}S_i S_k(1-\delta_{ik})\right) \end{aligned} \tag{5}$$

where S_i^2 and b_i^2 are the variances of the precision and bias error distributions respectively, $\rho_{S_{ik}}$ are the correlation coefficients for the precision errors, $\rho_{b_{ik}}$ are the correlation coefficients for the bias errors and δ_{ik} is the Kronecker delta defined to equal 1 when $i = k$ and 0 when $i \neq k$.

The uncertainty at some specified confidence level (such as a 95%) is thus defined as:

$$U_r = Ku_c \tag{6}$$

The choice of which coverage factor to use is discussed extensively in [1] Annexure 2-A. For this implementation it was assumed that the data has a Gaussian distribution and a confidence level of 95%. References [1] and [2] do indicate that the error distribution may often be considered Gaussian because of the Central Limit Theorem. Furthermore, they also indicate that for most practical wind-tunnel tests assuming a Gaussian distribution prevents the false sense of significance that may be attached to the computed numbers when using them. Also, the propagation equation is approximate (being a first order Taylor expansion only).

The 95% confidence expression for an uncertainty, U_r, is thus:

$$\begin{aligned} U_r^2 = & \sum_{i=1}^{j}\left(\left(\frac{\partial r}{\partial X_i}B_i\right)^2 + \right. \\ & \left. \sum_{k=1}^{j}\frac{\partial r}{\partial X_i}\frac{\partial r}{\partial X_k}\rho_{B_{ik}}B_iB_k(1-\delta_{ik})\right) + \\ & \sum_{i=1}^{j}\left(\left(\frac{\partial r}{\partial X_i}P_i\right)^2 + \right. \\ & \left. \sum_{k=1}^{j}\frac{\partial r}{\partial X_i}\frac{\partial r}{\partial X_k}\rho_{P_{ik}}P_iP_k(1-\delta_{ik})\right) \end{aligned} \quad (7)$$

2.1.1 Dependent Parameters

The partial derivatives of an uncertainty solution must be derived with respect to independent parameters [1]. A common error that is made in uncertainty analysis is to derive Equation 7 (i.e. the partial derivatives) using dependent parameters. This is because equations are normally derived using subsequent equations that are dependent on a number of independent parameters. For example, a result x is, in an analytical derivation, dependent on variables y and z.

$$x = x(y,z) \quad (8)$$

Variable y is dependent on the independent parameters f and g, and variable z is dependent on independent variables g and h.

$$\begin{aligned} y &= y(f,g) \\ z &= z(g,h) \end{aligned} \quad (9)$$

As both variables y and z are dependent on g, erroneous uncertainty results can be obtained if the uncertainty in x is derived with respect to y and z, and not f, g and h. This requirement results in analytically more difficult solutions, however.

As an example, consider the derivation of a coefficient C_l defined by the following formula:

$$C_l = \frac{M_x}{qSl_{ref}} \quad (10)$$

where M_x is a moment, q is the dynamic pressure, S is the reference area and l_{ref} is the reference length.

The dynamic pressure, q, is a function of total pressure, P_t, static pressure, P_s and the Mach number correction, ΔM.

$$q = f(P_t, P_s, \Delta M) \quad (11)$$

The uncertainty of C_l, ΔC_l, is thus a function of the uncertainties of the independent parameters, M_x, P_t, P_s and ΔM.

3 Generalised Uncertainty Analysis Scheme

3.1 Introduction

To determine the uncertainty equation of a parameter (for example a displacement parameter and in particular the state variables) for a store being simulated in a captive trajectory system is analytically difficult and error prone. This is because the variable for which the uncertainty is to be determined must be derived in terms of the system independent parameters before the partial derivatives can be derived. For a static test the derivation of the uncertainty equation for a single coefficient is also non-trivial. Furthermore, release constraint conditions are employed in CTS tests, which results in a dependent parameter having different independent parameters as the trajectory is being simulated. An example of this would be where a missile rail release is simulated. The first part of the release constrains the missile to one or two degrees-of-freedom. Once all except of the last missile shoe is free, the release constraints change to possibly a pivot or hook release. Only once the last shoe is free can the traditional six degree-of-freedom equations of motion be used to simulate the trajectory. For the constrained release modes, different equations of motion are used. To be as generic as possible, the generation of the uncertainty equations for all desired dependent parameters would need to be determined as the trajectory is being generated.

3.2 Requirements

The primary requirement for a generalised uncertainty analysis scheme is the ability to generate uncertainty equations as the trajectory is being simulated, because the independent parameters may change during the release simulation. This implies the need of the scheme to utilise the actual trajectory generation equations as the program is executed and not prior to coding or execution.

3.3 Proposed Solution

At least two methods can be used to meet the above requirements. The first utilises a jitter program, and the second uses symbolic mathematics.

A jitter program [2] utilises the data reduction algorithms as a subroutine and successively iterates using finite difference approximations to determine the partial derivatives necessary for the uncertainty analysis. The advantage of this solution is that it utilises the actual data reduction routines. Thus updating the data reduction routines results in the automatic updating of the uncertainty analysis process. This kind of solution would be feasible to implement for a trajectory simulation system. It does, however, require the use of successive iterations to determine the partial derivatives.

Symbolic mathematics is an elegant solution as the uncertainty equations for the parameters of interest would be to derived symbolically, and as functions of the system independent parameters only. The symbolic mathematics solution does not require successive iterations as that of the data reduction program, only a single pass, where all the partial derivatives are calculated at the end. This is achieved using the Maple computer algebra system, where the independent parameters are expressed as symbols and deriving subsequent variables or parameters from the independent parameters as defined by the data reduction process by executing the data reduction code. The scheme determines the equations as the program is executed, and not prior to program creation or execution. Thus changes to the trajectory generation code do not affect the process of determining the derived parameter equations because they are assessed during program execution. The determination of the uncertainty of any derived parameter is thus a straightforward operation once the derived parameter is expressed as a function of the system independent parameters.

The Maple computer algebra system kernel has been included in Matlab through the symbolic toolbox. The trajectory generation code has been coded in Matlab for code development purposes, with the production system coded in FORTRAN. The Matlab code base is used during the code development process to initially develop subsequent versions and as an implementation standard because the code structure and data flow logic are identical. This allows the generalised uncertainty analysis scheme to use the Matlab code base for the uncertainty scheme, the symbolic mathematics options being chosen for its elegance.

Unfortunately, the use of the actual trajectory generation equations to determine the equation of a derived parameter implies the use of comparison operations to determine code branching. Comparison operations are not possible if the symbols do not have a numerical value associated with them. This difficulty is overcome by the fact that the symbolic toolbox in Matlab has implemented a symbol variable as an object, thus allowing object extension.

3.4 Generalised Uncertainty Analysis Algorithm

The algorithm employed is shown in figure 1. The scheme is started by initialising the independent parameters. The precision and bias uncertainties and and correlation coefficients for the independent parameters are stored in a computer file. For each time step of the trajectory determination process, the uncertainties for the variables under consideration (at minimum the state variables) are determined symbolically. At the end of analysis process, the uncertainty values are simply calculated numerically from the symbolic equation. For a trajectory, because subsequent steps are dependent on the previous results, the uncertainties of the independent parameters of the next step must be updated.

Of particular importance is the updating of the uncertainty of the store loads (three orthogonal forces and three orthogonal moments) and positions (three orthogonal linear and three orthogonal rotational). This is because combined uncertainties result in the store not being positioned absolutely correctly. These uncertainties may result in a trajectory being significantly different to the ’true’ trajectory. This occurs when the gradients in interference loads are high. The differences between the ’true’ and simulated positions ultimately result in a simulated trajectory with a high uncertainty. This growing uncertainty is taken into by the algorithm increasing the uncertainty of the store loads in direct proportion to the store load gradient with respect to store position. These gradient data are obtained from grid tests. Thus for a store where the load gradient is zero, the store load or coefficient uncertainty will be that as obtained from the balance. For a store in a high gradient zone, the store load uncertainty is a combination of the balance uncertainty and the load gradient multiplied by the positional uncertainty.

The other area that required attention was the ordinary differential equation solver. The trajectory utilised a fifth order Adams-Moulton predictor-corrector alogrithm with a fifth order Runge-Kutta initializer. It was decided to simplify the propogation of errors by using a first order Euler method, rather than the Runge-Kutta Adams-Moulton alogrithm. While the use of an Euler algorithm is not recommended for the generation of the trajectory itself, because the uncertainty analysis is a truncated first order Taylor expansion, it was deemed sufficiently accurate to use the Euler algorithm for the uncertainty propogation. The state variable values (after each time step) used for the uncertainty equations were that as calculated by the trajectory generation code (fifth order Adams-Moulton predictor-corrector algorithm).

3.5 Matlab Implementation

3.5.1 *Matlab Objects*

A short discussion on the implementation of object oriented concepts in Matlab is pertinent before the details of the implementation are described. Matlab implements objects through classes i.e. an object is defined as a class (similar to fourth generation object oriented programming languages such as C++ and Java). These classes have similar capabilities to fourth generation object oriented programming languages such as C++ and Java. Of particular importance are the capabilities of polymorphism and encapsulation, both of which are exhibited by Matlab objects.

Classes or objects encapsulate object functions (or methods in Matlab) and object data (or fields in Matlab). Matlab objects (or base class) can be extended (polymorphism) through the creation of subclasses or derived classes, with subclasses thus inheriting methods and fields from its parent or base class, and adding or altering methods and/or fields.

3.5.2 *Implementation*

Implementing the object oriented concepts in the Matlab programming environment was performed by extending the base Matlab symbol object to include the numerical value of the symbol, thus allowing the comparison operations to be performed as the formulations of derived parameters are determined according to the implemented equations. The implementation of the generalised scheme would be considerably more complex without the object oriented capabilities of Matlab and its representation of symbols as objects.

A generalised uncertainty analysis scheme can thus be created which utilises the actual equations employed in the data reduction process. This is possible because of the following factors:

- The data reduction process is performed in Matlab

- Matlab can transparently use symbols and base type variables such as floating point numbers
- Matlab has object oriented capabilities

With reference to figure 1, the generalised uncertainty scheme first generates the data reduction equations of all subsequently derived parameters by defining the system independent parameters as symbols. Subsequent derived parameters are thus expressed in symbolic format as they are defined. This thus allows the uncertainty of any dependent parameter to be assessed using equation 7.

4 The Uncertainty Objects and their Implementation

The generalised uncertainty analysis scheme is based on two objects namely a parameter symbol and uncertainty variable. The parameter symbol, “usym” is a subclass of a Matlab symbol class, “sym”, while the uncertainty variable represents the uncertainty of the parameter under consideration. It would have been more elegant to encapsulate the uncertainty fields of a parameter in the parameter symbol. The separation of the uncertainty variable from the parameter symbol was required because the numerical calculation of the uncertainty was performed in a separate thread to speed execution.

4.1 Parameter Symbol

As mentioned previously, the parameter symbol class, “usym”, is a subclass of the Matlab symbol class that extends the Matlab class by including a field containing the numerical value of the symbol. As mentioned in section 3, a subclassed parameter symbol, “usym”, was required because the generalised uncertainty scheme uses the actual Matlab code to determine the symbolic form of any derived parameter. Any code branching because of comparison operators (such as greater than and less than in if statements) cannot be performed on non-numerical symbols. Implementing the parameter symbol requires the overloading of all mathematical functions that can be performed on symbols. These include:

- addition, subtraction, division and multiplication (both scalar and matrix)
- trigonometric operations such as sineous, cosineous and tangent
- comparison operations such as less than, and greater than

4.2 Uncertainty Variable

The uncertainty variable of a parameter is implemented as a class which encapsulates the following fields:

- uncertainty variable name
- numerical value
- independent parameters
- derivatives of the dependent variable with respect to the independent parameters
- precision and bias values
- correlation values (bias and precision)
- correlation coefficients (bias and precision)

5 Test Case

A test case was used to evaluate the uncertainty analysis scheme. The test case used was for a trajectory run performed on the 1/15th stable store from the NACA wing-body parent. The run release conditions were as follows:

- $M = 0.4$
- $P_t = 150kPa$
- $T_t = 310K$
- $dz = 10mm$
- Store Angle $= 0^o$
- Parent Angle $= 0^o$

- $C_a = 0.015$
- $W_i = 10 ft/s$

During the test run the only uncertainties assessed were the state variables. These were the body axes velocities (translational and rotational), body axes displacements, hook release body velocities and the body to inertial path axes transformation matrix (from which the rotational displacements can be obtained), i.e. 21 variables. Furthermore, no bias uncertainties and correlation limits were used, only precision uncertainties.

6 Results and Discussion

6.1 Results

The results of the test case are displayed in figures 2 to 7. The inertial referenced translational and rotational displacements are shown with the uncertainty bounds.

6.2 Jitter versus Symbolic Mathematics

The decision to use the symbolic mathematics option instead of the jitter program was driven by the elegance of the symbolic mathematics solution. The separate components contributing to the uncertainty of a parameter can be extracted not only numerically but symbolically, allowing greater analysis of its relative contribution. The use of a jitter program could, however, be easier due to its black box approach. The substitution of the Adams-Moulton ODE solver for the Euler solver would also not be required.

6.3 State Variable Uncertainties

The results obtained for the uncertainty analysis show a relatively large uncertainty for the rotational degrees of freedom, and in particular the pitch displacement, θ_I. The vertical displacement uncertainty is small, as expected, because the interference effect is small and the store is predominantly under the influence of gravity, which within the CTS code is considered a constant. The horizontal displacement, X_I, shows an increasing uncertainty even though the axial force was simulated through the use of a constant (see section 5) because an uncertainty bound was applied to the axial force coefficient.

6.4 Execution Speed and Implementation Efficiency

As mentioned previously (see section 4), the actual uncertainty code was separated into two separate threads to increase the execution speed. The generalised approach only facilitates a minimum level of optimisation. Furthermore, for each time step, the symbolic form of the equations are rederived. Optimisations can be made in this regard, especially when the store no longer has any applicable release constraints in the simulation.

7 Conclusions

A generalised uncertainty analysis scheme was devised for the captive trajectory system of the MSWT, CSIR. The scheme determines the uncertainty equations of derived parameters, using the trajectory generation equations, that are functions of system independent parameter only. The equations are derived as the program is executed. Implementing this scheme in Matlab was achieved using the symbolic toolbox with the Maple kernel and the subclassing of the symbol object to include the numerical value of the symbol or parameter so that comparison operations can be performed as the code is traversed. Parameters of interest are thus derived with respect to system independent parameters instead of derived parameters.

A test case using a trajectory run of the 1/15th scale stable store being released from the NACA wing-body model was used to assess the implementation. The results obtained were as expected. The current implementation can be optimised with respect to the algorithm that generates the uncertainty equations.

References

[1] American Institute of Aeronautics and Astronautics. *Assessment of Wind Tunnel Data Uncertainty*. AIAA S-071-1995.

[2] Coleman H.W. & Steele W.G. Jr. *Experimentation and Uncertainty Analysis for Engineers*. John Wiley and Sons, 2nd edition, 1989.

[3] American Institute of Aeronautics and Astronautics. *Assessing of Experimental Uncertainty - Supplement to AIAA S-071A-1999*. AIAAG-045-2003.

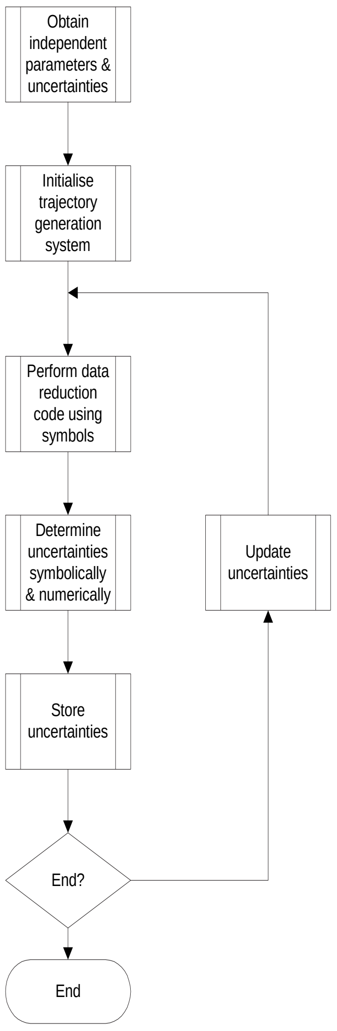
Obtain independent parameters & uncertainties
Initialise trajectory generation system
Perform data reduction code using symbols
Determine uncertainties symbolically & numerically
Update uncertainties
Store uncertainties
End?
End

Fig. 1 Generalised Uncertainty Analysis Scheme

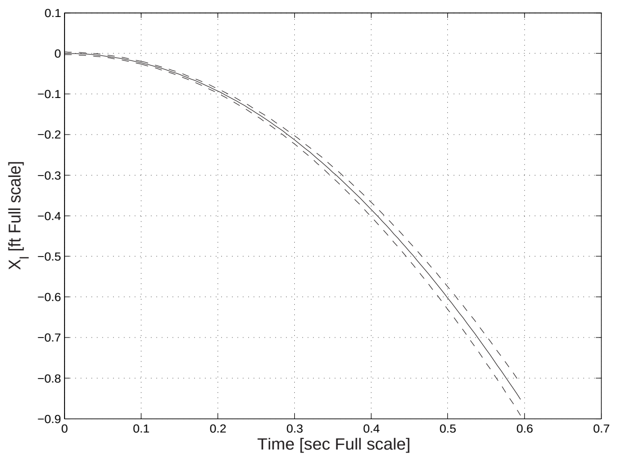

Fig. 2 Translational Displacement, X_I, as a Function of Time with Uncertainty Bounds

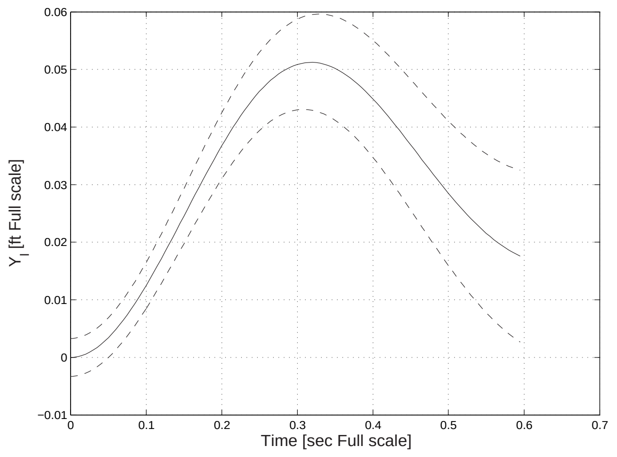

Fig. 3 Translational Displacement, Y_I, as a Function of Time with Uncertainty Bounds

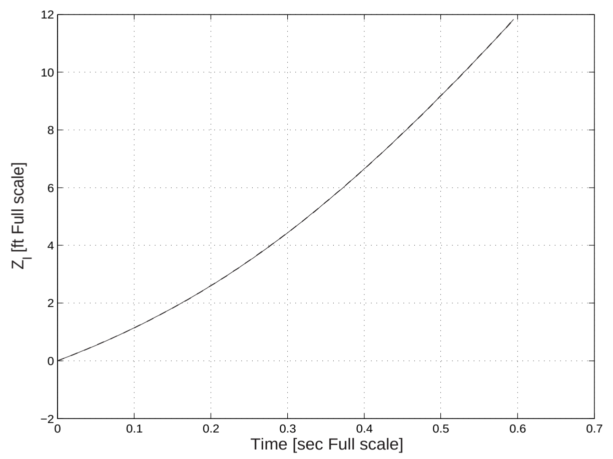

Fig. 4 Translational Displacement, Z_I, as a Function of Time with Uncertainty Bounds

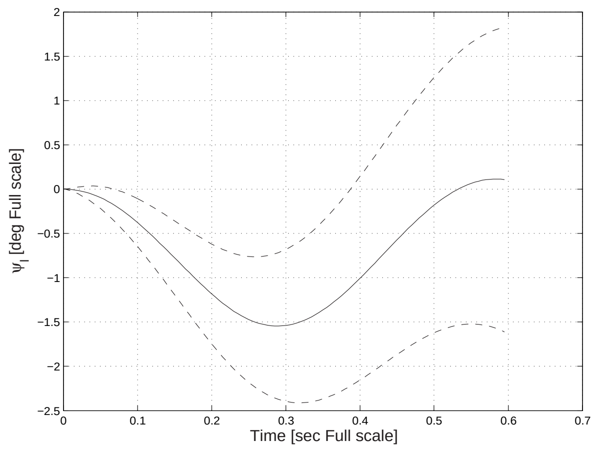

Fig. 6 Rotational Displacement, ψ_I, as a Function of Time with Uncertainty Bounds

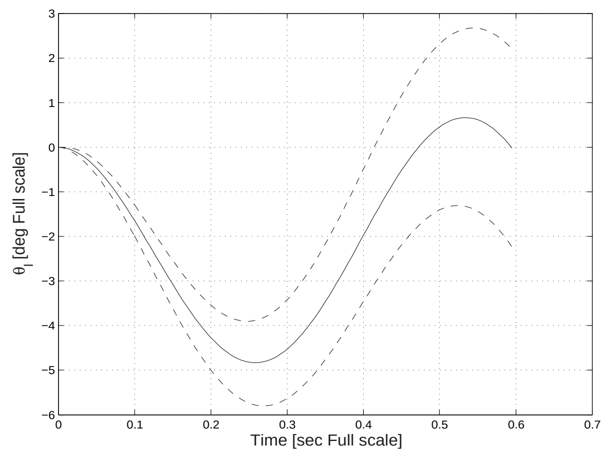

Fig. 5 Rotational Displacement, θ_I, as a Function of Time with Uncertainty Bounds

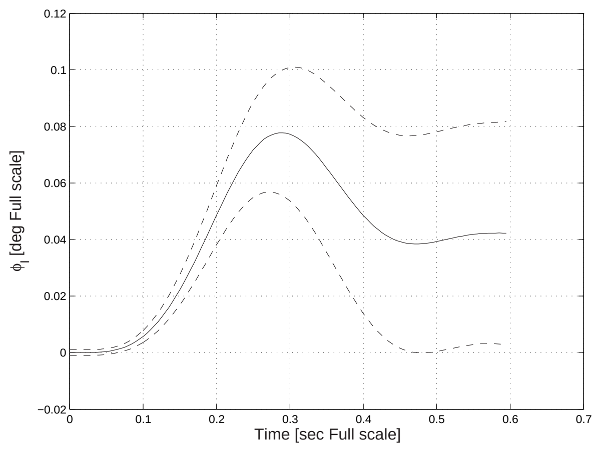

Fig. 7 Rotational Displacement, ϕ_I, as a Function of Time with Uncertainty Bounds