Draft-9 Manuscript Dated **Mar. 21, 2009**
Submitted to 2009 SIMULIA Customer Conf.
(Formerly ABAQUS Internat'l Users' Conf.),
May 18-21, 2009, London, England, U.K.
Ref.:
Draft_9_a9321_Fong_dMFH_UK_Conf.doc

A Design-of-Experiments Plug-In for Estimating Uncertainties in Finite Element Simulations (*)

Jeffrey T. Fong[1], Roland deWit[1], Pedro V. Marcal[2], James J. Filliben[1], and N. Alan Heckert[1]

[1]National Institute of Standards and Technology, Gaithersburg, MD 20899 U.S.A.
[2]MPave Corp, Julian, CA 92036 U.S.A.

Abstract: The objective of this paper is to introduce an economical and user-friendly technique for estimating a specific type of finite element simulation uncertainties, or, "error bars," for a class of mathematical models, of which no closed-form or approximate solution is known to exist. Using PYTHON, ABAQUS-PYTHON, and a public-domain statistical data analysis software package named DATAPLOT, we present a design-of-experiments-based PYTHON-DATAPLOT-Uncertainty Plug-In (PD-UP) to estimate the finite element simulation uncertainties in a mathematical model due to variability in modeling parameters such as material property constants, geometric parameters, loading rates, etc. Examples of finite element simulations of the free vibrations of a single-crystal silicon cantilever beam in an atomic force microscope with an estimation of the uncertainties of one of the simulation results, namely, the first bending mode resonance frequency of the beam, are included.

Keywords: ABAQUS, ANSYS, Atomic Force Microscope, DATAPLOT, Design of Experiments, Finite Element Method, Mechanics, Nanotechnology, Plug-In, PYTHON, Simulation, Single Crystal Silicon, Statistical Data Analysis, Uncertainty Analysis, Verification, Vibration.

1. Introduction

It is well-known that simulation results of the finite element method (FEM) contain errors due to a variety of sources (see, e.g., Hughes 1987, Haldar 1997, Zienkiewicz 2000, Chu 2002, Yang 2002, Lord 2003, Fong 2005, Fong 2006a, Fong 2006b, Fong 2008a). By and large, such errors are of major concern to experimentalists or modelers, when the governing equations and their associated material property constants, boundary conditions, geometric parameters, and loading rates, etc., are known to contain significant uncertainties as defined for physical measurements in ISO 1993 and Taylor 1994, and for black-box computer models by a recent paper of Kennedy 2001. In this paper, we address FEM simulation errors by going beyond Kennedy 2001 in the sense that we no longer treat an FEM model as a black-box. Using a design-of-experiments approach and a 10-step statistical analysis algorithm (Filliben 2002), we develop a plug-in for an FEM model with an uncertainty estimation and a robustness ranking of not only the parametric variables but also the FEM code platforms (e.g., ABAQUS vs. ANSYS) and FEM mesh sizes (coarse vs. fine).

2. Statistical Theory of Design of Experiments (abbrev. DOE)

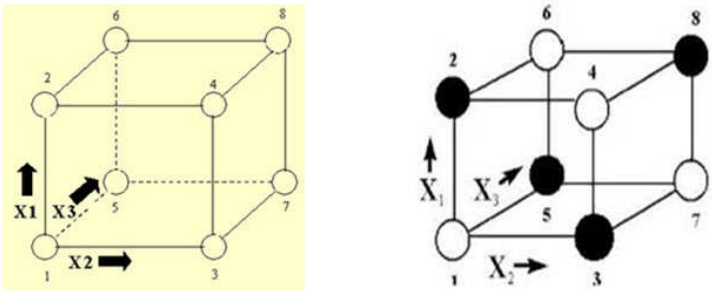

Figure 1. (left) **A full-factorial 8-run orthogonal design for 3 factors.**
(right) **A fractional factorial 4-run orthogonal design for 3 factors.**

To estimate finite element simulation uncertainties due to physical modeling, we propose to do a virtual experiment by changing one or more physical process variables (factors) in order to observe the effect the changes have on one or more response variables. In an earlier paper (Fong 2008a), we propose a DOE-based methodology (Box 1978, Montgomery 2000), because DOE is an efficient procedure for planning experiments so that the data obtained can be analyzed to yield valid and objective conclusions with respect to the relative importance of the factors.

DOE begins with determining the *objectives* of an experiment and selecting the *process factors* for the study. An Experimental Design is the laying out of a detailed experimental plan in advance of doing the experiment. Well chosen experimental designs maximize the amount of "information" that can be obtained for a given amount of experimental effort. The statistical theory underlying DOE begins with the concept of *process models*. A process *model* of the 'black box' type is formulated with several discrete or continuous input *factors* that can be controlled, and one or more measured output *responses.* The output responses are assumed continuous. Real or virtual experimental data are used to derive an empirical (approximate) model linking the outputs and inputs. These empirical models generally contain *first-order (linear) and second-order (quadratic and interactions) terms.* For a complete exposition of this topic, readers are referred to the articles cited in the reference section of this paper (Box 1978, Montgomery 2000, Croarkin 2003, Fong 2008a).

The most popular and cost-effective experimental designs for sensitivity analysis are two-level designs. To illustrate the concept, we show in Fig. 1 (left) a graphical representation of a two-level, full factorial design for three factors, namely, the 2^3 design, and Fig. 4 (right) the same of a two-level, fractional factorial design, or, the 2^{3-1} design. The arrows show the direction of increase of the factors in order to represent the user-defined variability of each factor. The numbers "1" through "8" at the corners of the design box refer to the "Standing Order" of runs (also known in the literature as the "Yates Order").

3. A DOE for an FEM Computer Model with a Known Solution

In Table 1, we present the results of an FEM computational experiment (Fong 2008a) for a classical problem in mechanics where the first bending resonance frequency, q_1, of an isotropic elastic cantilever beam of rectangular cross-section is known (Timoshenko 1955, Clough 2003). This makes it possible for us to design a 3-factor, full-factorial computer experiment with two platforms (ABAQUS vs. ANSYS) and two mesh sizes (4 for coarse, and 12 for fine). Using a 10-step analysis of the DOE data (Filliben 2002), we show in Fig. 2 a plot of the ranking of the importance of the three factors as well as a estimated uncertainty plot of q_1 for the ABAQUS coarse mesh (m = 4) run. In Fig. 3, we show a comparison of the ABAQUS runs for two mesh sizes and found, as expected, the finer mesh (m = 12) to be closer to the exact solution. In Fig. 4, we add two more factors to the computer experiment by including the ANSYS runs in order to answer the question (Fig. 5 using block plot) whether the mesh size effect is "real."

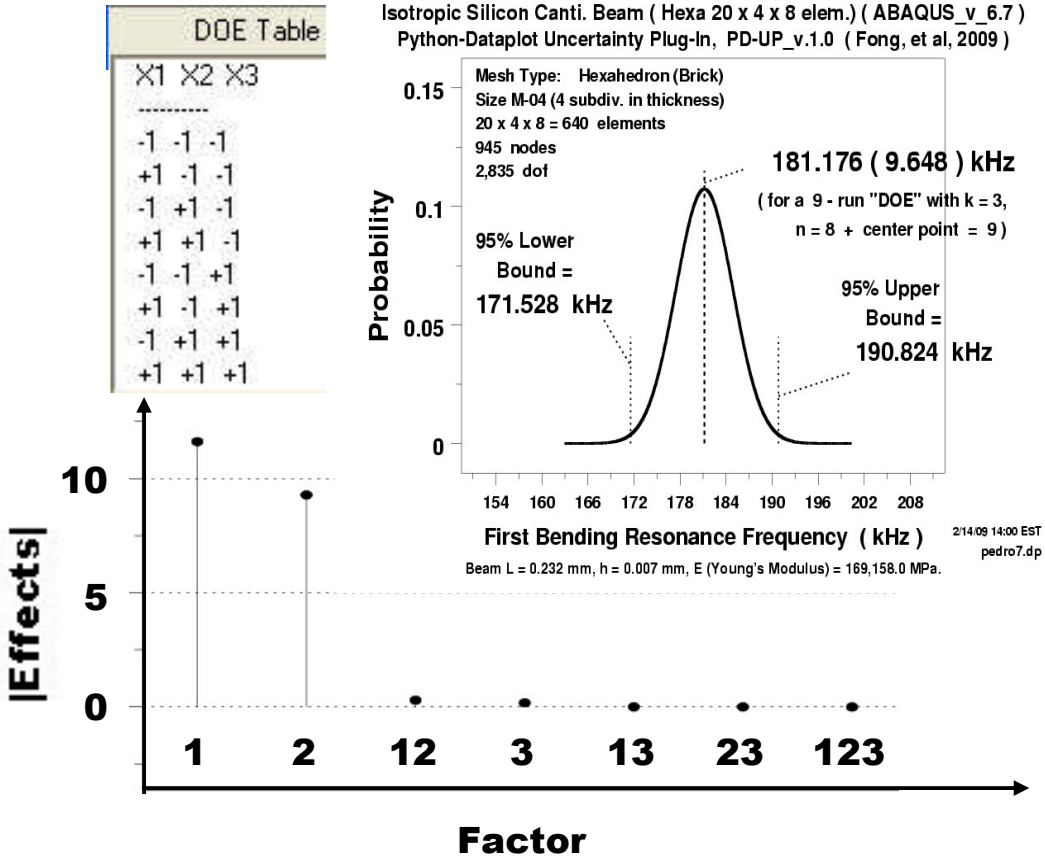

Figure 2. A ranking of main effects plot and an uncertainty estimate plot.

3. A DOE for an FEM Computer Model with a Known Solution (Continued)

Table 1. FEM Computer Modeling of First Bending Resonance Frequency q1 (kHz) of an Isotropic Elas. Cantilever Beam in an Atomic Force Microscope (Fong 2008a)

3-factor (L, h, E), full factorial DOE	Exact Theoretical Solution	Abaqus Mesh No. m = 4	Ansys Mesh No. m = 4	Abaqus Mesh No. m = 12	Ansys Mesh No. m = 12
(0, 0, 0)	179.026	181.052	181.238	180.503	180.523
(-1, -1 ,-1)	180.049	182.119	182.306	181.566	181.587
(+1 ,-1,-1)	168.891	170.807	170.981	170.286	170.305
(-1 ,+1,-1)	189.553	191.692	191.888	191.113	191.135
(+1,+1,-1)	177.806	179.786	179.970	179.241	179.261
(-1,-1,+1)	180.229	182.301	182.488	181.748	181.768
(+1,-1,+1)	169.060	170.977	171.152	170.456	170.475
(-1,+1,+1)	189.742	191.883	192.080	191.304	191.326
(+1,+1,+1)	177.984	179.966	180.150	179.421	179.441

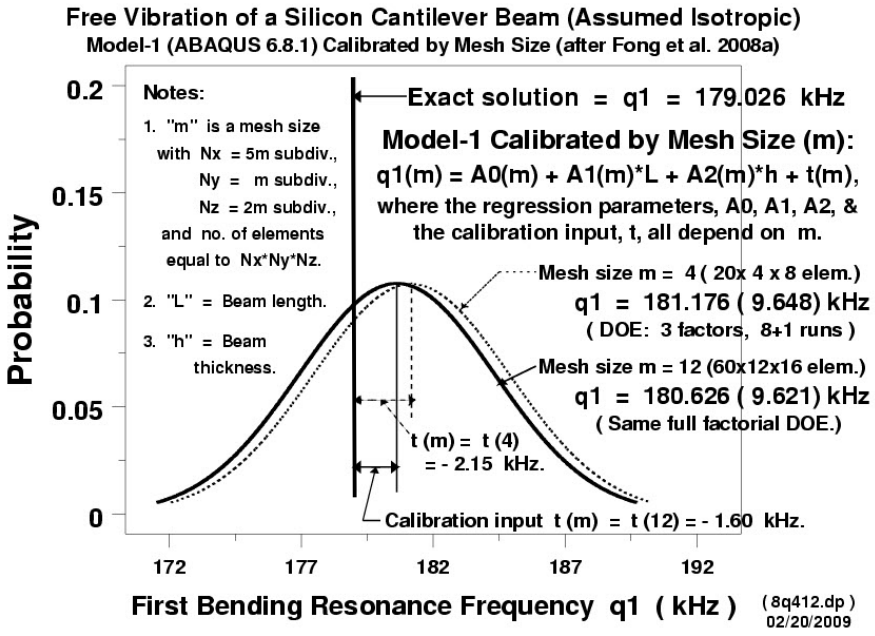

Figure 3. An FEM model showing effect of mesh size from coarse to fine

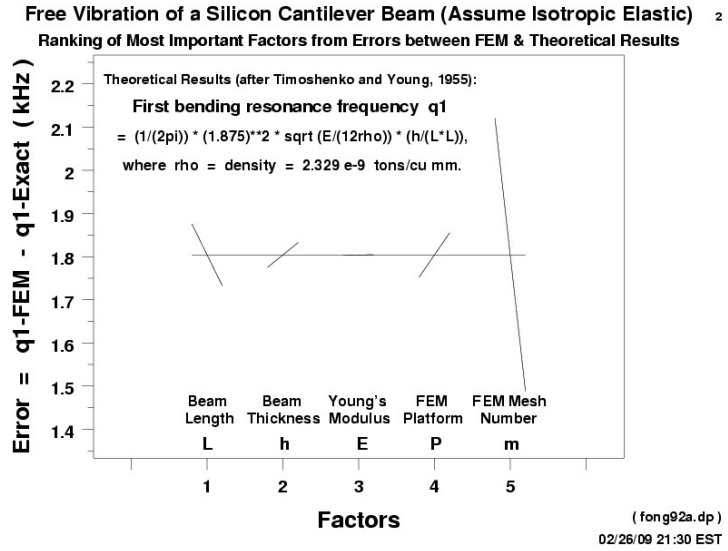

Figure 4. A ranking of five factors based on a DOE analysis of errors.

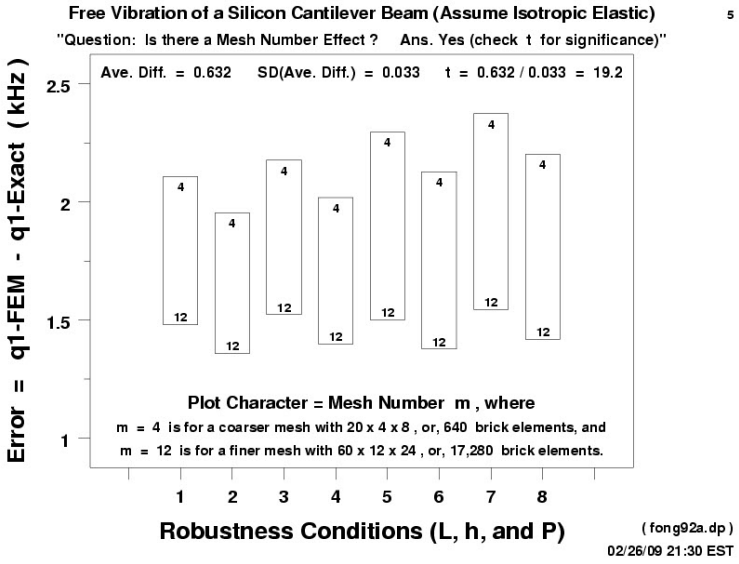

Figure 5. A robustness analysis of FEM data for mesh size effect.

4. A Conceptual Design of a Python-Dataplot Uncertainty Plug-In

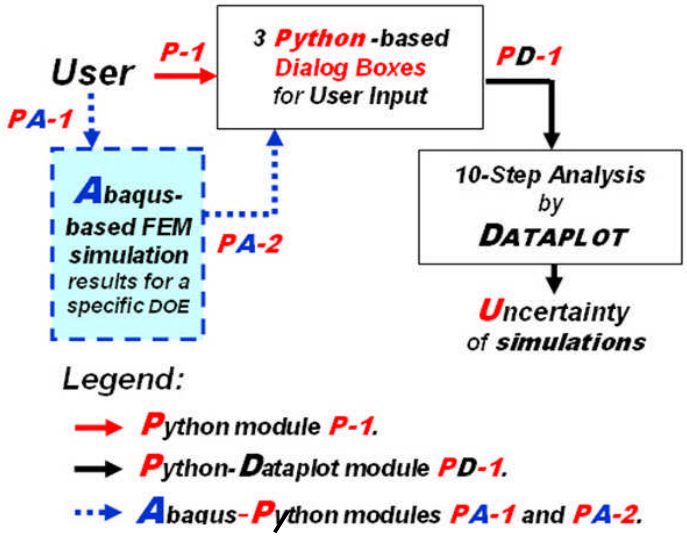

Figure 6. A Conceptual Design of the Python-Dataplot-Uncertainty Plug-In.

Throughout the FEM experiment described in the last section, we were burdened with the task of repeatedly solving the same FEM problem with a different set of parameters prescribed by a DOE. This motivated us to design a computational plug-in to facilitate an FEM uncertainty estimation methodology. As shown in Fig. 6, the so-called Python-Dataplot-Uncertainty Plug-In (abbrev. PD-UP) consists of two separate parts, namely, an FEM part and an Uncertainty Estimation part:

(1) The FEM part, which can be implemented in any packages such as ABAQUS, ANSYS, LSDYNA, etc., involves the design of two modules, one of which, PA-1, is simply to make a single base run of the FEM model, and the other, PA-2, to modify the input file of PA-1 for several more runs according to a design of experiments (DOE) as specified by the user (see, e.g., Box 1978 and Montgomery 2000).

(2) The Uncertainty Estimation part, which is implemented in this paper in PYTHON (van Rossum 1999, Harms 2000, Hammond 2000, Deitel 2005) and a public domain package named DATAPLOT (Filliben, 2002, Croarkin 2003), also involves the design of two modules, one of which, P-1, is to receive input from the user with two dialog boxes, Button_1 and Button_2, and the other, PD-1, to review the input data with the user in Button_3 before the user clicks a box to execute a 10-step analysis according to a user-defined design of experiments.

Since the second part involves PYTHON, it is advantageous for us to implement the first part with ABAQUS for which the ABAQUS-PYTHON interface is readily available and facilitates the complete automation of a finite element uncertainty estimation tool as described in Fong 2008a.

5. A Two-Button Design for User's Input Module, P-1

The user's input module, **P-1**, consists of two dialog boxes, **Button_1** (*PyDpGui_1.py*) and **Button_2** (*PyDpGui_2.py*). A complete listing of *PyDpGui_1.py* is given in a website and is downloadable.

The user is asked 5 questions in **Button_1** (Fig. 7, top) and 3 in **Button_2** (Fig. 7, bottom). The key question, (Q-1.1), in **Button_1** is the number of factors, k, in a two-level factorial design of experiments the user wishes to pursue. The other four questions are:

(Q-1.2) the name of an output file,

(Q-1.3) the labels and symbols of the k factors,

(Q-1.4) the values of the k factors for a base run of the finite element simulations (to be known later as the center point of the two-level hypercube design), and

(Q-1.5) the percent variability of each factor in a two-level DOE.

After the value of k is specified in **Button_1**, the user is asked in **Button_2** another key question, (Q-2.1), namely, n, the number of simulations the user wishes to run with a two-level factorial experimental design. If the user chooses a full factorial design, then $n = 2^k$. If one wishes to conduct an experiment with less than n runs, one may choose a fractional factorial design by selecting a finite number from the set, n_i, $i = 1, 2, \ldots p$, where $n_1 = 2^{k-1}$, $n_2 = 2^{k-2}$, . . . , $n_p = 2^{k-p}$, until the smallest n_p that is still greater than k. For example, if k = 5, one may choose 2^5 (= 32) runs for a full factorial design, or, either 2^4 (= 16), or, 2^3 (= 8) runs for a fractional factorial design, but not 2^2 (= 4) runs, because 4, being less than 5 (= k), violates the condition that the number of runs must be greater than the number of factors being considered (Box 1978). The other two questions in **Button_2** are:

(Q-2.2) the same name of the output file used in the answer for question Q-1.2 to insure continuity, and

(Q-2.3) the label and symbol of the response variable that is being computed from a finite element model involving those k factors and their variability.

At this point, the FEM part of the plug-in is activated (see **PA-1** and **PA-2** in Fig. 6) and the results of ($n+1$) simulations for a user-defined (k, n) design of experiments can be extracted from the FEM runs and stored as input to **Button_2** for uncertainty analysis.

5. A Two-Button Design for User's Input Module P-1 (Continued)

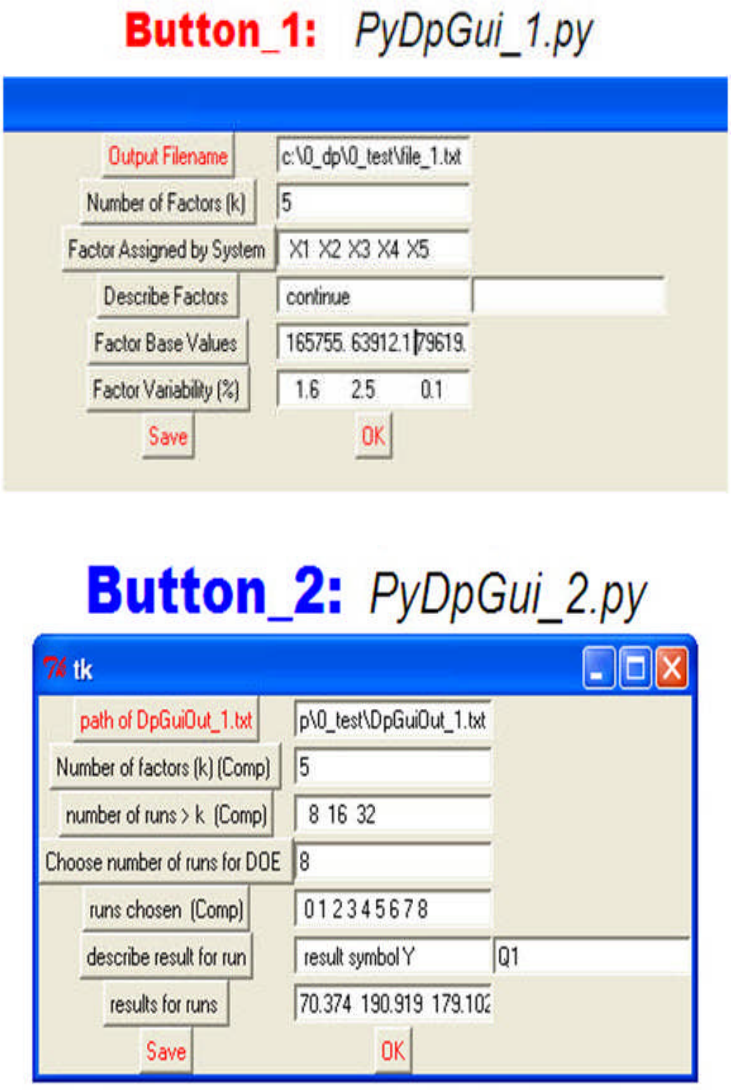

Figure 7. (Top) **Button_1 to define *k* factors and enter their variability**
(Bottom) **Button_2 to choose *n* runs for a (*k, n*) design and enter results.**

6. Design of Button_3, the Python-Dataplot Module, PD-1

We are now ready to activate the uncertainty estimation of the PD-UP design by invoking **Button_3** (PyDpGui_3.py. See Fig. 8.). The user is first asked to fill in the name of the output file specified earlier in **Button_1**. When that name is correct, three screens of input data will be displayed for review. The first screen (upper left of Fig. 8) is a review of the user's input to **Button_1**. The second screen (upper right of Fig 8) is a review of the user's input to **Button_2**. The third screen (lower left of Fig. 8) is new to the user, and contains the DOE table for the values of k and n furnished by the user in **Button_1** and **Button_2**.

If the user is not satisfied, the dialog box will return to **Button_1** and **Button_2**. Otherwise, a dataplot code named "*pedro7.dp*" is activated to run the 10-step statistical analysis and return an 11-plot file for viewing and downloading. The plot 11 is the uncertainty estimation result such as those given in Figs. 2 and 3.

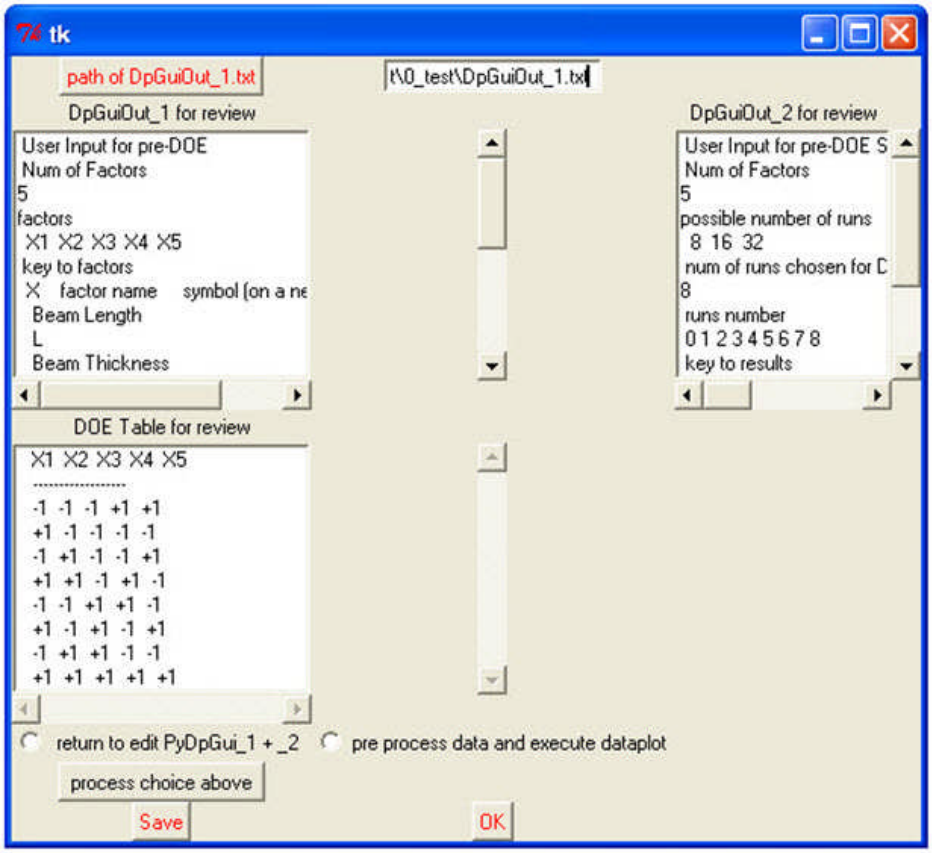

Figure 8. Button_3 with two of three output screens.

7. An Example of an FEM Model without a Known Solution

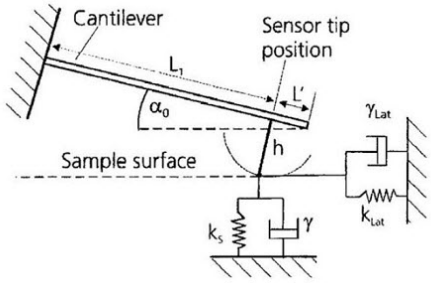

Figure 9. A cantilever beam inside an atomic force microscope (after Kester 2000).

In experiments (Rabe 1996, Hurley 2003, and Fig. 9 after Kester 2000) and computational models (Timoshenko 1955, Lekhnitskii 1981, and Fig. 10 after Fong 2006b) on the resonance frequencies of a single-crystal silicon cantilever beam in an atomic force microscope, Hurley 2003 and Fong 2008a reported a major discrepancy on the two lowest natural bending resonance frequencies of such beam in two different shapes, rectangular and dagger. For our purposes here, it suffices to introduce that well-documented problem as an example of a direct application of the newly developed Python-Dataplot-Uncertainty Plug-in. Using measurement variability data of single-crystal silicon elastic constants first published by McSkimin 1966, and geometric parameter variability data such as beam length and thickness commonly known to laboratory researchers, we propose a two-level, (k = 5, n = 8), fractional factorial orthogonal DOE with the five factors being the beam length (X1), the beam thickness (X2), the cubic-crystal silicon elastic constant C_{11} (X3), the second constant C_{12} (X4), and the third constant C_{44} (X5). The response variable is the first bending resonance frequency of a silicon cantilever beam in an atomic force microscope.

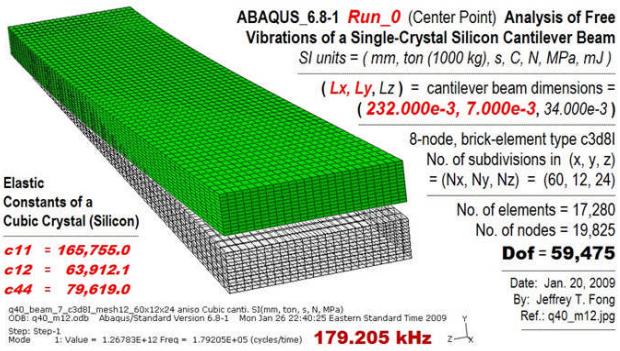

Figure 10. A typical ABAQUS simulation of free vibration of a cantilever beam.

8. Virtual Experimental Data According to an Orthogonal Design

As cited in Fong 2008a, the availability of a public-domain statistical data analysis package named DATAPLOT and the documentation (Croarkin 2003) of a 10-step exploratory data analysis (EDA) approach to the analysis of 2^k full factorial and 2^{k-p} fractional factorial designs, provided us an opportunity to fully develop an FEM uncertainty estimation tool as the analysis of a screening problem.. In general, there are two characteristics of a screening problem: (a) There are many factors to consider, all of equal *a priori* interest for the experiment at hand. (b) Each of these factors may be either continuous or discrete. The desired output from the analysis of a screening problem consists of: (i) A ranked list (by order of importance) of factors. (ii) The best settings for each of the factors. (iii) A good model. (iv) Most importantly, insight into the underlying mechanisms of the real process. When we use **Button_3** (*PyDpGui_3.py*) to estimate finite element uncertainties, we first obtain 10 informative plots (see Fong 2008a for a listing of those 10 plots), one of which, as shown in Fig. 11, is a plot on the ranking of the main effects as well as interactions among k factors ($k = 5$ in this example). We then used a linear regression fit of the two dominant factors (X1 and X2 in this case) to obtain an uncertainty estimate of the response as plot in the upper right corner of Fig. 11. Such a computational plug-in is quite useful when we deal with a non-simple real-life process such as the modeling of the resonance frequency of a single-crystal silicon beam in an atomic force microscope.

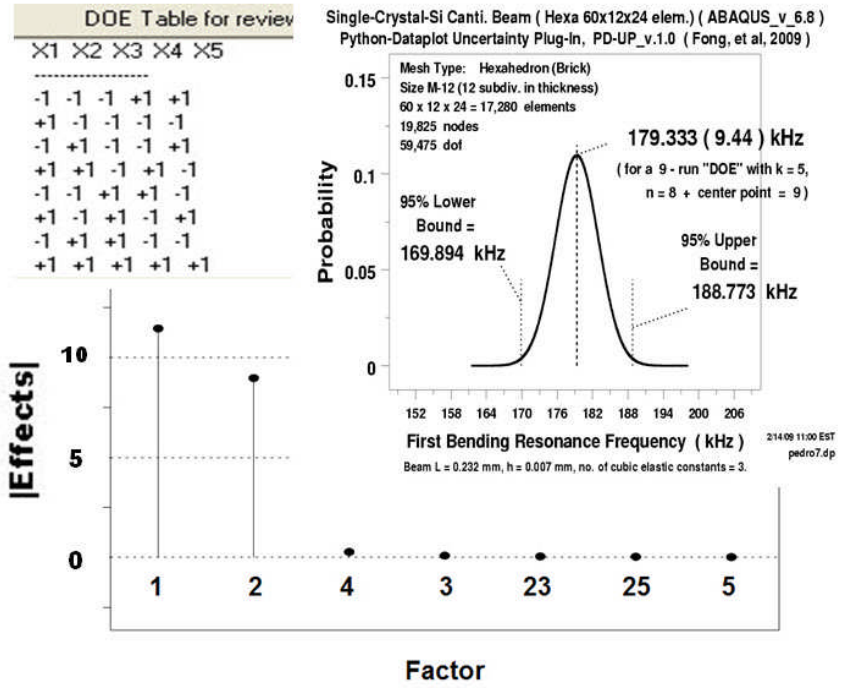

Figure 11. A main effects plot and an uncertainty estimate from a PD-UP

9. Discussion and Concluding Remarks

In this expository paper on a two-part code implementation of a DOE-based FEM uncertainty estimation methodology, we have presented only the uncertainty estimation part of the plug-in, namely, three PYTHON codes, *PyDpGui_1.py*, *PyDpGui_2.py*, and *PyDpGui_3.py*, and one DATAPLOT code, *pedro7.dp*. Note that all four codes are made available for downloading at http://math.nist.gov/mcsd/software.html. We have left the FEM part open, because our purpose is to make the plug-in, PD-UP (v. 1.0), available for any finite element uncertainty estimation effort using either commercially-available, proprietary, or public-domain FEM packages. For instance, when the FEM package is ABAQUS, we plan to specialize our PD-UP with a link to ABAQUS-PYTHON and name the resulting plug-in the ABAQUS-PD-UP. A similar link can be made to another PYTHON-based FEM package named MPACT, and the resulting plug-in will be named MPACT-PD-UP, and so on. All of the new plug-ins, of course, will be coded in PYTHON.

As illustrated in our numerous examples, we conclude that it is feasible to apply a DOE-based methodology to the estimation of FEM uncertainty due not only to model inadequacy, physical parameters, and observation errors, but also to computer black box parameters such as mesh size, type, and code implementation. Recent FEM simulations with or without using PYTHON as a scripting language (see, e.g., Langer 2001, Easley 2007, Chao 2008, Fong 2008b) and new FEM simulations such as Fong 2009 may benefit from this plug-in when uncertainty of the computer code vs. real process observations is a compelling issue.

Based on the results presented in Figs. 4 and 5 of Section 3, where we showed two ways of answering a question whether a specific factor such as the mesh size has an effect, we observe that the computational plug-in, PD-UP, v. 1.0, reported in this paper, is applicable only to the automation of the 10-step sensitivity and uncertainty analysis of FEM calculations (Fig. 4). A future verstion of PD-UP incorporating the block plot methodology (Fig. 5) as well as the remaining FEM part of the plug-in will complete the larger goal of this paper, namely, a cost-effective and user-friendly FEM uncertainty estimation for modeling real processes of multiple factors.

10. References

1. ABAQUS, "ABAQUS Scripting User's Manual**,** Version 6.8." Dassault Systemes Simulia Corp., 166 Valley Street, Providence, RI, 2008.
2. ABAQUS, "ABAQUS User's Manual**,** Version 6.8.0." Dassault Systemes Simulia Corp., 166 Valley Street, Providence, RI, 2008.
3. ANSYS, "ANSYS User's Manual**,** Release 10.0." ANSYS, Inc., 275 Technology Dr., Cannonsburg, PA, 2006.
4. Box, G. E., Hunter, W. G., and Hunter, J. S., "Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building." Wiley (1978).

5. Chao, Y. J., Fong, J. T., and Lam, P. S., "A New Approach to Assessing the Reliability of Applying Laboratory Fracture Toughness Test Data to Full-Scale Structures," Proc. ASME Pressure Vessels & Piping Conference, July 27-31, 2008, Chicago, IL, Paper No. PVP2008-61584, http://www.asmeconferences.org/PVP08, 2008.

6. Chu, E., Zhang, L., Wang, S., Zhu, X, and Maker, B., 2002, "Validation of Springback Predictability with Experimental Measurements and Die Compensation for Automotive Panels," Proc. 5th International Conf. and Workshop on Numerical Simulation of 3-D Sheet Metal Forming Processes - Verification of Simulation with Experiment, Jeju Islands, Korea, Oct. 21-25, 2002, Yang, D., Oh, S. I., Huh, H., and Kim, Y. H., eds., Vols. 1. pp. 313-318. Published by the Korean Adv Inst Sci & Tech., 373-1, Science Town, Taejon, 305-701, Korea 2002.

7. Clough, R., and Penzien, J., "Dynamics of Structures," 2nd ed. (Revised). Berkeley, CA 94704: Computers and Structures, Inc., 2003.

8. Croarkin, C., Guthrie, W., Heckert, N. A., Filliben, J. J., Tobias, P., Prins, J., Zey, C., Hembree, B., and Trutna, eds., 2003, "NIST/SEMATECH e-Handbook of Statistical Methods, Chapter 5 on Process Improvement (pp. 1-480)," http://www.itl.nist.gov/div898/handbook/, first issued, June 1, 2003, and last updated July 18, 2006. Produced jointly by the Statistical Engineering Division of the National Institute of Standards & Technology, Gaithersburg, MD, and the Statistical Methods Group of SEMITECH, Austin, TX. Also available as a NIST Interagency Report in a CD-ROM upon request to alan.heckert@nist.gov, 2006.

9. Deitel, H. M., Deitel, P. J., Liperi, J. P., and Wiedermann, B. A., "Python: How to Program," first edition. Prentice-Hall, 2005.

10. Draper, N. R., and Smith, H., "Applied Regression Analysis." Wiley (1966).

11. Easley, S., Pal, S., Tomaszewski, P., Petrella, A., Rulkowtter, P., and Laz, P., "Finite Element-based Probabilistic Analysis Tool for Orthopaedic Applications," Computer Methods and Programs in Biomedicine, Vol. 85, Issue 1, pp. 32-40, 2007.

12. Filliben, J. J., and Heckert, N. A., "DATAPLOT: A Statistical Data Analysis Software System," A Public Domain Software Released by NIST, Gaithersburg, MD 20899, http://www.itl.nist.gov/div898/software/dataplot.html, 2002.

13. Fong, J. T., "ABC of Statistics for Verification and Validation (V&V) of Simulations of High-Consequence Engineering Systems," Proc. 2005 ASME Pressure Vessels and Piping Conference, July 17-21, 2005, Denver, CO, Paper No. PVP2005-MF-13-1, 2005.

14. Fong, J. T., Filliben, J. J., deWit, R., Fields, R. J., Bernstein, B., and Marcal, P. V., "Uncertainty in Finite Element Modeling and Failure Analysis: A Metrology-Based Approach," ASME Trans., J. Press. Vess. Tech., Vol. 128, pp. 140-147, 2006a.

15. Fong, J. T., Filliben, J. J., deWit, R., and Bernstein, B., "Stochastic Finite Element Method (FEM) and Design of Experiments for Pressure Vessel and Piping (PVP) Decision Making," Proc. of 2006 ASME Pressure Vessels and Piping Division Conference, July 23-27, 2006, Vancouver, B. C., Canada, paper no. PVP2006-ICPVT11-93927. New York, NY: American Society of Mechanical Engineers, 2006b.

16. Fong, J. T., Filliben, J. J., Heckert, N. A., and deWit, R., "Design of Experiments Approach to Verification and Uncertainty Estimation of Simulations Based on Finite Element Method,"

Proc. Conf. Amer. Soc. for Engineering Education, June 22-25, Pittsburgh, PA, Paper AC 2008-2725, 2008a.

17. Fong, J. T., Ranson, W. F., III, Vachon, R. I., and Marcal, P. V., "Structural Aging Monitoring via Web-based Nondestructive Evaluation Technology," Proc. ASME Pressure Vessels & Piping Conference, July 27-31, 2008, Chicago, IL. Paper No. PVP2008-61607, http://www.asmeconferences.org/PVP08, 2008b.

18. Fong, J. T., deWit, R., Kim, Y. S., Dagalakis, N., Filliben, J. J., and Heckert, N. A., "A Metrological Approach to Comparing Accuracy of Finite Element Software Packages using Tetrahedral vs. Hexahedral Element Meshes," Manuscript in preparation, 2009.

19. Haldar, A., Guran, A., and Ayuub, B. M., eds. "Uncertainty Modeling in Finite Element, Fatigue and Stability of Systems." World Scientific Publishing Co. Pte. Ltd., 1060 Main Street, River Edge, NJ, 1997.

20. Hammond, M., and Robinson, A., "Python Programming on Win32," O'Reilly Media, Inc., 2000.

21. Harms, D., and McDonald, K., "The Quick Python Book," http://www.manning.com, Manning Pub. Co., 2000.

22. Hughes, T. J. R., "The Finite Element Method: Linear Static and Dynamic Finite Element Analysis." Prentice-Hall, 1987.

23. Hurley, D. C., Shen, K., Jennett, N. M., and Turner, J. A., "Atomic force acoustic microscopy methods to determine thin-film elastic properties," J. Appl. Phys., 94(4), 2347-2354, 2003.

24. ISO, 1993, "Guide to the Expression of Uncertainty in Measurement," Prepared by ISO Technical advisory Group 4 (TAG 4), Working Group 3 (WG 3), Oct. 1993. ISO/TAG 4, Sponsored by the BIPM (Bureau International des Poids et Mesures), IEC (International Electrotechnical Commission), IFCC (International Federation of Clinical Chemistry), ISO, IUPAC (Int. Union of Pure and Applied Chemistry), IUPAP (International Union of Pure and Applied Physics), and OIML (Int. Organization of Legal Metrology), 1993.

25. Kennedy, M. C., and O'Hagan, A., "Bayesian calibration of computer models," J. Royal Statistical Soc., Serial B, Vol. 63, Part 3, pp. 425-464 (2001).

26. Kester, E., Rabe, U., Presmanes, L., Tailhades, Ph., and Arnold, W., "Measurement of Young's Modulus of Nanocrystalline Ferrites with Spinel Structures by Atomic Force Acoustic microscopy," J. Phys. Chem. Solids, 61, 1275-1284, 2000.

27. Langer, S. A., Fuller, E. R., and Carter, W. C., "OOF: An Image-Based Finite-Element Analysis of Material Microstructure," Computing in Science and Engineering, Vol. 3, No. 3, pp. 15-23, May/June 2001.

28. Langtangen, H. P., "Python Scripting for Computational Science," third edition. Springer, 2008.

29. Lekhnitskii, S. G., "Theory of Elasticity of an Anisotropic Body," Translated from the revised 1977 Russian edition. Moscow: MIR Publishers, 1981.

30. Lord, G. J., and Wright, L., "Uncertainty Evaluation in Continuous Modeling," Report to the National Measurement System Policy Unit, Department of Trade and Industry, NPL Report CMSC 31/03. Teddington, Middlesex, U.K.: National Physical Laboratory, 2003.

31. LSTC, "LS-DYNA Keyword User's Manual, Version 970," April 2003, Livermore Software Technology Corp., Livermore, CA, 2003.

32. Marcal, P. V., "MPACT User's Manual." Published by MPACT Corp., Julian CA, marcalpv@cox.net, 2008.

33. McSkimin, H. J., and Andreatch, P., Jr., "Elastic Moduli of Silicon vs. Hydrostatic Pressure at 25.0 C and - 195.8 C," J. Appl. Phys., 35(7), 2161-2165, 1964.

34. Montgomery, D. C., "Design and Analysis of Experiments," 5th ed. Wiley, 2000.

35. Rabe, U., Janser, K., and Arnold, W., "Vibrations of free and surface-coupled atomic force microscope cantilevers: Theory and experiment," Rev. Sci. Instrum., 67 (9), 3281-3293, 1996.

36. Taylor, B. N., and Kuyatt, C. E., "Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results," NIST Tech. Note 1297, Sep. 1994 edition (supersedes Jan. 1993 edition), Prepared under the auspices of the NIST Ad Hoc Committee on Uncertainty Statements, U. S. Government Printing Office, Washington, DC, 1994.

37. Timoshenko, S., and Young, D. H., "Vibration Problems in Engineering," 3rd ed. D. Van Nostrand, 1955.

38. van Rossum, G., "Python Tutorial," Feb. 19, 1999, Release 1.5.2. Corp. for National Research Initiative (CNRI), 1895 Preston White Dr., Reston, VA, 1999.

39. Yang, D., Oh, S. L., Huh, H., and Kim, Y. H., eds., "Numisheet 2002: Design Innovation Through Virtual Manufacturing," Proc. 5th Int. Conf. and Workshop on Numerical Simulation of 3D Sheet Forming Processes - Verification of Simulation with Experiment, 21–25 October 2002, Jeju Island, Korea, Vol. 2, published by Korean Adv Inst Sci & Tech., 373-1, Science Town, Taejon, 305-701, Korea, 2002.

40. Zienkiewicz, O. C., and Taylor, R. L., "The Finite Element Method," 5th ed., Vol. 1, The Basis. Butterworth-Heinemann, 2000.

11. Acknowledgment

We wish to thank Ron Boisvert, Andrew Dienstfrey, and Alden Dima, all of NIST, Andrew MacKeith and David C. Winkler of Dassault Systemes Simulia Corp., and Robert Rainsberger of XYZ Scientific Applications, Inc., Livermore, CA, for their technical discussions/comments during the course of this investigation.

12. Disclaimer

The views expressed in this paper are strictly those of the authors and do not necessarily reflect those of their affiliated institutions. The mention of the names of all commercial vendors and their products is intended to illustrate the capabilities of existing products, and should not be construed as endorsement by the authors or their affiliated institutions.

This page is left blank.

Appendix A

Listing of Python Code for Button_1

PyDpGui_1.py

(Also available on line at http://math.nist.gov/mcsd/software.html.)

```
#! /usr/bin/env python


import string
import cPickle
import os
import re
from Tkinter import *
class App:
    def __init__(self, master):
        self.m_label=" to be set"
        self.m_entry5=False
        frame = Frame(master)
        frame.pack()
        self.m0 = Button(frame, text="Output File", fg="red",
command=self.InverseProgram0)
        self.m0.grid(row=0,column=0)
        self.e0 =Entry(frame)
        columnspan=2
        self.e0.grid(row=0,column=1 )
        #self.m1.grid(row=1,column=0)
        self.m1 = Button(frame, text="Number of factors (k) ",
command=self.readInEntry1)
        self.m1.grid(row=1,column=0)
        self.e1=Entry(frame)
        self.e1.grid(row=1,column=1)
        self.m2 = Button(frame, text="Factors ",
command=self.readInEntry2)
        self.m2.grid(row=2,column=0)
        self.e2=Entry(frame)
        self.e2.grid(row=2,column=1)
        self.m5 = Button(frame, text="Describe Factors ",
command=self.readInEntry5)
        self.m5.grid(row=3,column=0)
```

```
        self.e5=Entry(frame)
        self.e5.grid(row=3,column=1)
        self.e6=Entry(frame)
        self.e6.grid(row=3,column=2)
        self.m3 = Button(frame, text="Center point values ",
command=self.readInEntry3)
        self.m3.grid(row=4,column=0)
        self.e3=Entry(frame)
        self.e3.grid(row=4,column=1)
        self.m4 = Button(frame, text="Variability (%) ",
command=self.readInEntry4)
        self.m4.grid(row=5,column=0)
        self.e4=Entry(frame)
        self.e4.grid(row=5,column=1)

        self.button1 = Button(frame, text="Save", fg="red",
command=self.InverseProgram1)
        self.button1.grid(row=6,column=0)
        self.button2 = Button(frame, text="OK", fg="red",
command=frame.quit)
        self.button2.grid(row=6,column=1)
    def readInEntry1(self):
        keyWord=self.e1.get()
        try :
            self.m_factors=string.atoi(keyWord)
        except :
            self.m_factors=-1
    def readInEntry2(self) :
        line=' '
        for i in range (self.m_factors) :
            line+=' X'+`i+1`+' '
        try :
            self.e2.insert(0,line)
        except :
            pass
        line+='\n'
        self.m_factorsLabel=line
        self.m_label=' press this button to start '
        self.e5.delete(0,END)
        self.e5.insert(0,self.m_label)
    def readInEntry5(self) :
        if not self.m_entry5 :
            self.m_entry5=True
            self.m_readFactor=0
            self.m_readText=True
            self.m_factorDescription=[]
```

```
            self.m_factorInput=False
        else :
            pass
        if self.m_readFactor<self.m_factors :
            if self.m_readText :
                if not self.m_factorInput :
                    line = ' factor name X'+`self.m_readFactor+1`
                    self.m_label=line
                    self.e5.delete(0,END)
                    self.e5.insert(0,self.m_label)
                    self.m_factorInput=True
                else :
                    keyWord = self.e6.get()
                    self.m_factorDescription.append(keyWord)
                    self.m_readText=False
                    self.m_factorInput=False
                    self.e6.delete(0,END)
                    line = ' factor symbol
X'+`self.m_readFactor+1`
                    self.m_label=line
                    self.e5.delete(0,END)
                    self.e5.insert(0,self.m_label)
                    self.m_factorInput=True
            else :
                if not self.m_factorInput :
                    pass
                else :
                    keyWord = self.e6.get()
                    self.m_factorDescription.append(keyWord)
                    self.m_readText=True
                    self.m_factorInput=False
                    self.e6.delete(0,END)
                    line = ' continue '
                    self.m_label=line
                    self.e5.delete(0,END)
                    self.e5.insert(0,self.m_label)
                    self.m_readFactor+=1
        else :
            self.m_entry5=False
            self.m_label=" end of description"
            self.e5.delete(0,END)
            self.e5.insert(0,self.m_label)
    def readInEntry3(self):
        keyWord=self.e3.get()
        self.m_centerPointString=keyWord+'\n'
        fields=string.split(keyWord)
```

```
        flen=len(fields)
        centerPoint=[]
        for i in range(flen) :
            centerPoint.append(string.atof(fields[i]))
        self.m_centerPoint=centerPoint
    def readInEntry4(self):
        keyWord=self.e4.get()
        self.m_variabilityString=keyWord+'\n'
        fields=string.split(keyWord)
        flen=len(fields)
        variability=[]
        for i in range(flen) :
            variability.append(string.atof(fields[i]))
        self.m_variability=variability
    def InverseProgram0(self) :
        keyWord=self.e0.get()

        self.m_concept={}
        self.m_concept["basePath"]=keyWord
    def InverseProgram1(self) :
        self.m_FileName = self.m_concept["basePath"]
        dir_name=os.path.dirname(self.m_FileName)
        self.m_FileName=os.path.basename(self.m_FileName)
        dir_name=os.path.join(dir_name,self.m_FileName)
        try :
            filer=open(dir_name,'w')
            line=' Output from pre DOE \n'
            filer.write(line)
            filer.flush()
            line=' Num of Factors \n'
            filer.write(line)
            line=`self.m_factors`+'  \n'
            filer.write(line)
            filer.flush()

            filer.write('factors   \n')
            filer.write(self.m_factorsLabel)
            filer.flush()
            filer.write(' key to factors \n')
            filer.write('  X    factor name     symbol (on a
newline) \n')
            count=-2
            for i in range (self.m_factors) :
                count+=2
                line=' '+'  '+self.m_factorDescription[count]+'
\n'
```

```
                filer.write(line)
                line='   '+self.m_factorDescription[count+1]+ '
\n'
                filer.write(line)
            filer.write(' center Point values \n')
            filer.write(self.m_centerPointString)
            filer.flush()
            filer.write(' variability (%) \n')
            filer.write( self.m_variabilityString)
            filer.flush()
        except :
            pass
        filer.close()
root = Tk()
app = App(root)
root.mainloop()
```

This page is left blank.

Appendix B

Listing of Python Code for Button_2

PyDpGui_2.py

(Also available on line at http://math.nist.gov/mcsd/software.html.)

```
#! /usr/bin/env python


import string
import cPickle
import os
import re
from Tkinter import *
class App:
    def __init__(self, master):
        self.m_entry6=False
        frame = Frame(master)
        frame.pack()
        self.m0 = Button(frame, text="path of DpGuiOut_1.txt",
fg="red", command=self.InverseProgram0)
        self.m0.grid(row=0,column=0)
        self.e0 =Entry(frame)
        self.e0.grid(row=0,column=1 )
        #self.m1.grid(row=1,column=0)
        self.m1 = Button(frame, text="Number of factors (k)
(Comp) ", command=self.readInEntry1)
        self.m1.grid(row=1,column=0)
        self.e1=Entry(frame)
        self.e1.grid(row=1,column=1)
        self.m2 = Button(frame, text="number of runs > k
(Comp)", command=self.readInEntry2)
        self.m2.grid(row=2,column=0)
        self.e2=Entry(frame)
        self.e2.grid(row=2,column=1)
        self.m3 = Button(frame, text="Choose number of runs for
DOE ", command=self.readInEntry3)
        self.m3.grid(row=3,column=0)
```

```
        self.e3=Entry(frame)
        self.e3.grid(row=3,column=1)
        self.m4 = Button(frame, text="runs chosen  (Comp)",
command=self.readInEntry4)
        self.m4.grid(row=4,column=0)
        self.e4=Entry(frame)
        self.e4.grid(row=4,column=1)
        self.m6 = Button(frame, text="describe result for run",
command=self.readInEntry6)
        self.m6.grid(row=5,column=0)
        self.e6=Entry(frame)
        self.e6.grid(row=5,column=1)
        self.e7=Entry(frame)
        self.e7.grid(row=5,column=2)
        self.m5 = Button(frame, text="results for runs ",
command=self.readInEntry5)
        self.m5.grid(row=6,column=0)
        self.e5=Entry(frame)
        self.e5.grid(row=6,column=1)

        self.button1 = Button(frame, text="Save", fg="red",
command=self.InverseProgram1)
        self.button1.grid(row=7,column=0)
        self.button2 = Button(frame, text="OK", fg="red",
command=frame.quit)
        self.button2.grid(row=7,column=1)
    def readInEntry1(self):
        keyWord=self.e1.insert(0,`self.m_factors`)
    def readInEntry2(self) :
        count=-1
        prod=1
        product=[]
        for k in range(self.m_factors) :
            prod=prod*2
            if prod<= self.m_factors :
                continue
            product.append(prod)
        klen=len(product)
        if klen >4 :
            klen=4
        line=' '
        for i in range (klen) :
            line+=' '+`product[i]`+' '
        try :
            self.e2.insert(0,line)
        except :
```

```
            pass
        line+='\n'
        self.m_factorsRuns=product
        self.m_factorsRunsString=line

    def readInEntry3(self):
        keyWord=self.e3.get()
        try :
            self.m_factorsChosen=string.atoi(keyWord)
        except :
            self.m_factorsChosen=-1
    def readInEntry4(self):
        line=' '
        sum=[]
        for k in range(self.m_factorsChosen+1) :
            line+=`k`+' '
            sum.append(k)
        try :
            self.e4.insert(0,line)
        except :
            pass
        self.m_chosenRuns=sum
        self.m_chosenRunsString=line+'\n'
        self.m_label=' press this button to start '
        self.e6.delete(0,END)
        self.e6.insert(0,self.m_label)
        self.m_ReadResults=0
    def readInEntry5(self):
        keyWord=self.e5.get()
        self.m_resultsPointString=keyWord+'\n'
        fields=string.split(keyWord)
        flen=len(fields)
        resultsPoint=[]
        for i in range(flen) :
            resultsPoint.append(string.atof(fields[i]))
        self.m_resultsPoint=resultsPoint
    def readInEntry6(self) :
        if not self.m_entry6 :
            self.m_entry6=True
            self.m_readFactor=0
            self.m_readText=True
            self.m_resultDescription=[]
            self.m_factorInput=False
        else :
            pass
        if self.m_ReadResults< 1 :
```

```
            if self.m_readText :
                if not self.m_factorInput :
                    line = ' result name Y'
                    self.m_label=line
                    self.e6.delete(0,END)
                    self.e6.insert(0,self.m_label)
                    self.m_factorInput=True
                else :
                    keyWord = self.e7.get()
                    self.m_resultDescription.append(keyWord)
                    self.m_readText=False
                    self.m_factorInput=False
                    self.e7.delete(0,END)
                    line = ' result symbol Y'
                    self.m_label=line
                    self.e6.delete(0,END)
                    self.e6.insert(0,self.m_label)
                    self.m_factorInput=True
            else :
                if not self.m_factorInput :
                    pass
                else :
                    keyWord = self.e7.get()
                    self.m_resultDescription.append(keyWord)
                    self.m_readText=True
                    self.m_factorInput=False
                    self.e7.delete(0,END)
                    line = ' end of description '
                    self.m_label=line
                    self.e6.delete(0,END)
                    self.e6.insert(0,self.m_label)
                    self.m_ReadResults+=1
                    self.m_entry6=False
    def InverseProgram0(self) :
        keyWord=self.e0.get()

        self.m_concept={}
        self.m_concept["basePath"]=keyWord
        self.m_FileName = self.m_concept["basePath"]
        dir_name=os.path.dirname(self.m_FileName)
        self.m_FileName=os.path.basename(self.m_FileName)
        dir_name=os.path.join(dir_name,self.m_FileName)
        try :
            filer=open(dir_name,'r')
            self.m_lines=filer.readlines()
            flen=len(self.m_lines)
```

```
        except :
            pass
        filer.close()
        line=self.m_lines[2]
        fields=string.split(line)
        self.m_factors=string.atoi(fields[0])

    def InverseProgram1(self) :
        self.m_FileName = self.m_concept["basePath"]
        dir_name=os.path.dirname(self.m_FileName)
        self.m_FileName='DpGuiOut_2.txt'
        dir_name=os.path.join(dir_name,self.m_FileName)
        try :
            filer=open(dir_name,'w')
            line=' Output from pre DOE \n'
            filer.write(line)
            filer.flush()
            line=' Num of Factors \n'
            filer.write(line)
            line=`self.m_factors`+'  \n'
            filer.write(line)
            filer.flush()

            filer.write('num of runs   \n')
            filer.write(self.m_factorsRunsString)
            filer.flush()
            filer.write(' num of runs chosen for DOE \n')
            filer.write(`self.m_factorsChosen`+' \n')
            filer.flush()
            filer.write(' runs number\n')
            filer.write( self.m_chosenRunsString)
            filer.flush()
            filer.write(' key to results \n')
            filer.write('  Y    result name     symbol (next
line) \n')
            count=-2
            for i in range (1) :
                count+=2
                line=' '+'  '+self.m_resultDescription[count]+ '
\n'
                filer.write(line)
                line=' '+self.m_resultDescription[count+1]+ '
\n'
                filer.write(line)
            filer.write(' results for runs\n')
            filer.write( self.m_resultsPointString)
```

```
            filer.flush()
        except :
            pass
        filer.close()
root = Tk()
app = App(root)
root.mainloop()
```

Appendix C

Listing of Python Code for Button_3

PyDpGui_3.py

(Also available on line at http://math.nist.gov/mcsd/software.html.)

```
#! /usr/bin/env python


import string
import cPickle
import os
import re
from Tkinter import *
import sys
from subprocess import call

class App:
    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        self.m0 = Button(frame, text="path of DpGuiOut_1.txt",
fg="red", command=self.InverseProgram0)
        self.m0.grid(row=0,column=0)
        self.e0 =Entry(frame)
        self.e0.grid(row=0,column=1 )
        #self.m1.grid(row=1,column=0)
        self.yScrol=Scrollbar(frame,orient=VERTICAL )
        self.yScrol.grid(row=2, column=1, sticky=N+S)
        self.xScrol=Scrollbar(frame,orient=HORIZONTAL )
        self.xScrol.grid(row=3, column=0, sticky=E+W )
        self.listbox = Listbox
(frame,xscrollcommand=self.xScrol.set,
             yscrollcommand=self.yScrol.set , exportselection =
0 )
        self.listbox.grid (row=2 ,column=0,sticky=N+S+E+W )
        self.xScrol["command"] = self.listbox.xview
        self.yScrol["command"] = self.listbox.yview
        self.yScroll=Scrollbar(frame,orient=VERTICAL )
```

```
        self.yScroll.grid(row=2, column=4, sticky=N+S)
        self.xScroll=Scrollbar(frame,orient=HORIZONTAL )
        self.xScroll.grid(row=3, column=3, sticky=E+W )
        self.listb = Listbox
(frame,xscrollcommand=self.xScroll.set,
              yscrollcommand=self.yScroll.set , exportselection =
0 )
        self.listb.grid (row=2 ,column=3,sticky=N+S+E+W )
        self.xScroll["command"] = self.listb.xview
        self.yScroll["command"] = self.listb.yview
        self.m2 = Label(frame, text=" DpGuiOut_1 for review " )
        self.m2.grid(row=1,column=0)
        self.m3 = Label(frame, text=" DpGuiOut_2 for review " )
        self.m3.grid(row=1,column=3)

        self.yScrolt=Scrollbar(frame,orient=VERTICAL )
        self.yScrolt.grid(row=5, column=1, sticky=N+S)
        self.xScrolt=Scrollbar(frame,orient=HORIZONTAL )
        self.xScrolt.grid(row=6, column=0, sticky=E+W )
        self.listbtx = Listbox
(frame,xscrollcommand=self.xScrolt.set,
              yscrollcommand=self.yScrolt.set , exportselection =
0 )
        self.listbtx.grid (row=5 ,column=0,sticky=N+S+E+W )
        self.xScrolt["command"] = self.listbtx.xview
        self.yScrolt["command"] = self.listbtx.yview
        self.m4 = Label(frame, text=" DOE Table for review " )
        self.m4.grid(row=4,column=0)

        self.v=IntVar()
        self.radButton1=Radiobutton(frame,text=' return to edit
PyDpGui_1 + _2 ',variable=self.v,value=1)
        self.radButton1.grid(row=7,column=0)
        self.radButton2=Radiobutton(frame,text='pre process data
and execute dataplot',variable=self.v,value=2)
        self.radButton2.grid(row=7,column=1)

        self.m5 = Button(frame, text=" process choice above ",
command=self.readInEntry3)
        self.m5.grid(row=8,column=0)
        self.button1 = Button(frame, text="Save", fg="red",
command=self.InverseProgram1)
        self.button1.grid(row=9,column=0)
        self.button2 = Button(frame, text="OK", fg="red",
command=frame.quit)
        self.button2.grid(row=9,column=1)
```

```
    def readInEntry3(self):
        v=self.v.get()
        if v==1 :
            pass
        if v==2 :
            returncode=None
            try :
                returncode= call('C:/Program
Files/NIST/DATAPLOT/dataplot < pedro7.dp' )
                pass
                if returncode :
                    print 'Failure with return code' ,returncode
            except :
                pass
            i=1

os.spawnl(os.P_NOWAIT,'C:/Ghostgum/gsview/gsview32.exe','
C:/0_____dataplot/0_test/DPPL1F.DAT')

    def InverseProgram0(self) :
        keyWord=self.e0.get()

        self.m_concept={}
        self.m_concept["basePath"]=keyWord
        self.m_FileName = self.m_concept["basePath"]
        dir_name=os.path.dirname(self.m_FileName)
        self.m_FileName=os.path.basename(self.m_FileName)
        dir_name=os.path.join(dir_name,self.m_FileName)
        try :
            filer=open(dir_name,'r')
            self.m_lines=filer.readlines()
            flen=len(self.m_lines)
        except :
            pass
        filer.close()
        line=self.m_lines[2]
        fields=string.split(line)
        self.m_factors=string.atoi(fields[0])
        for line in self.m_lines :
            lines=string.replace(line,'\n','')
            self.listbox.insert(END,lines)

        self.m_FileName = self.m_concept["basePath"]
        dir_name=os.path.dirname(self.m_FileName)
        self.m_FileName='DpGuiOut_2.txt'
```

```
        dir_name=os.path.join(dir_name,self.m_FileName)
        try :
            filer=open(dir_name,'r')
            self.m_line=filer.readlines()
            flen=len(self.m_line)
        except :
            pass
        filer.close()
        readVars=True
        readOnce=False
        count=-1
        for line in self.m_line :
            count+=1
            lines=string.replace(line,'\n','')
            self.listb.insert(END,lines)
            tes=string.find(line,'runs')
            if tes >= 0 and not readOnce :
                if readVars :
                    readVars=False
                else :
                    readVars=True

                if readVars :
                    self.m_numRuns=self.m_line[count+1]
                    a=self.m_numRuns
                    as=string.split(a)
                    self.m_numRuns=string.atoi(as[0])
                    readOnce=True
        spaceRun=4
        spaceFactors=2
        fileName=''
        if self.m_factors < 10 :
            fileName+='0'+`self.m_factors`
        else :
            fileName+=`self.m_factors`
        if self.m_numRuns < 10 :
            fileName+='000'+`self.m_numRuns`
        elif self.m_numRuns >=10 and self.m_numRuns < 100 :
            fileName+='00'+`self.m_numRuns`
        elif self.m_numRuns >=100 and self.m_numRuns < 1000 :
            fileName+='0'+`self.m_numRuns`
        elif self.m_numRuns >=1000 and self.m_numRuns < 10000 :
            fileName+=`self.m_numRuns`

fileName+='_k='+`self.m_factors`+'_and_n='+`self.m_numRuns`+'_DOE
_table.txt'
```

```
        self.m_FileName = 'C:/0_____dataplot/0_test/doe_tables'
        dir_name=self.m_FileName
        self.m_FileName=fileName
        dir_name=os.path.join(dir_name,self.m_FileName)
        try :
            filer=open(dir_name,'r')
            self.m_line=filer.readlines()
            flen=len(self.m_line)
        except :
            pass
        filer.close()
        self.m_FileName = self.m_concept["basePath"]
        dir_name=os.path.dirname(self.m_FileName)
        self.m_FileName='pedro7.txt'
        dir_name=os.path.join(dir_name,self.m_FileName)
        try :
            filer=open(dir_name,'w')
        except :
            pass
        for line in self.m_line :
            filer.write(line)
        filer.close()
        xLabels=False
        for line in self.m_line :
            a=string.split(line)
            alen=len(a)
            if alen >=2 and not xLabels :
                if a[0]=='X1' and a[1]=='X2' :
                    xLabels=True
            if xLabels :
                lines=string.replace(line,'\n','')
                self.listbtx.insert(END,lines)


    def InverseProgram1(self) :
        self.m_FileName = self.m_concept["basePath"]
        dir_name=os.path.dirname(self.m_FileName)
        self.m_FileName='DpGuiOut_3.txt'
        dir_name=os.path.join(dir_name,self.m_FileName)
        # END NO OUTPUT TO SAVE
root = Tk()
app = App(root)
root.mainloop()
```

This page is left blank.

Appendix D

Listing of DATAPLOT Code named

pedro7.dp

(Also available on line at http://math.nist.gov/mcsd/software.html.)

```
. ---------------------------------------------------------------------
.
. DATAPLOT program:  pedro7.dp OR pedro7_a90226_k3_n8+1_Exact.dp
.                    ---------    ---------------------------------
.
.             for a two-level full or fractional factorial orthogonal
.             design of experiments (DOE) analysis automated by a
.             3-button PYTHON-script Plug-In named PD-UP (version 1.0).
.
. Authors:    Jeffrey Fong, NIST, Gaithersburg MD    fong@nist.gov
.             Roland deWit, NIST, Gaithersburg MD    dewit@nist.gov
.             Pedro Marcal, MPACT Corp., Julian CA    marcalpv@cox.net
.             James Filliben, NIST, Gaithersburg MD   filliben@nist.gov
.      and N. Alan Heckert, NIST, Gaithersburg MD  alan.heckert@nist.gov
.
. Date Initiated: Jan. 13, 2009.        Revised: Feb. 26, 2009 3:00 EST
.
. ---------------------------------------------------------------------
.
.                              Limitations:
.     No. of factors (k) to vary:                        from 2 to 10.
.     No. of runs (n) for fractional designs to vary:    from 4 to 128.
.     No. of runs (n) for full factorial DOE to vary:    from 4 to 1024.
.
. ---------------------------------------------------------------------
.
.  System Requirements:
.
.  3 text files created by PD-UP Python codes and read by this program:
.
.           DpGuiOut_1.txt,   DpGuiOut_2.txt,   pedro7.txt
.
.  Dataplot code, embedded in DATAPLOT subdirectory as installed
.          in C:\Program Files\NIST, and called by this program:
.
.          dexplot.dp   (No need to have it in working directory)
.
.  32 tables of DOE based on Reference [1] cited below (Box, Hunter,
.          Hunter, p.410, Table 12.15) must be included as text files
.          in a subdirectory named "doe_tables" and installed in the
.          user's working directory to be read by a Python code:
.
.          020004_k=2_and_n=4_DOE_table.txt
```

```
.
.          030004_k=3_and_n=4_DOE_table.txt
.          030008_k=3_and_n=8_DOE_table.txt
.
.          040008_k=4_and_n=8_DOE_table.txt
.          040016_k=4_and_n=16_DOE_table.txt
.
.          050008_k=5_and_n=8_DOE_table.txt
.          050016_k=5_and_n=16_DOE_table.txt
.          050032_k=5_and_n=32_DOE_table.txt
.
.          060008_k=6_and_n=8_DOE_table.txt
.          060016_k=6_and_n=16_DOE_table.txt
.          060032_k=6_and_n=32_DOE_table.txt
.          060064_k=6_and_n=64_DOE_table.txt
.
.          070008_k=7_and_n=8_DOE_table.txt
.          070016_k=7_and_n=16_DOE_table.txt
.          070032_k=7_and_n=32_DOE_table.txt
.          070064_k=7_and_n=64_DOE_table.txt
.          070128_k=7_and_n=128_DOE_table.txt
.
.          080016_k=8_and_n=16_DOE_table.txt
.          080032_k=8_and_n=32_DOE_table.txt
.          080064_k=8_and_n=64_DOE_table.txt
.          080128_k=8_and_n=128_DOE_table.txt
.          080256_k=8_and_n=256_DOE_table.txt
.
.          090016_k=9_and_n=16_DOE_table.txt
.          090032_k=9_and_n=32_DOE_table.txt
.          090064_k=9_and_n=64_DOE_table.txt
.          090128_k=9_and_n=128_DOE_table.txt
.          090512_k=9_and_n=512_DOE_table.txt
.
.          100016_k=10_and_n=16_DOE_table.txt
.          100032_k=10_and_n=32_DOE_table.txt
.          100064_k=10_and_n=64_DOE_table.txt
.          100128_k=10_and_n=128_DOE_table.txt
.          101024_k=10_and_n=1024_DOE_table.txt
.
. ---------------------------------------------------------------------
.
.  Output:  Plots 1 to 10 (for Analysis 10 steps) as postscript files
.                         for viewing and downloading using Ghostview.
.
.           Plot 11 (95% Confidence Bounds) also as postscript file.
.
.           One text file named "pedro7_UNC1.TXT" for user to present
.           a (n+1)-run 2-parameter fit results and a table of 95%
.           uncertainty bound results.
.
. ---------------------------------------------------------------------
.
.  Application-specific Plot Notes:
.
.     4 Plot notes are provided to flag the user to make changes
.                 in this program to suit individual application.
.
```

```
.     In Plot Note-1:   Define "plcolor" for plot color.
.
.     In Plot Note-2:   Define "currdate" for current date and time.
.
.     In Plot Note-3:   Define "finelem" for the finite element code name
.                                        and version number.
.                       Define "project" for heading in each of 11 plots.
.                       Define "xxlabel" for Plot 11 xlabel.
.
.     In Plot Note-4:   Define "bxlabel" for a footnote below Plot 11 xlabel.
.
.                       Define "annodof1" for Mesh Type annotation in Plot 11.
.                       Define "annodof2" for Mesh Size annotation in Plot 11.
.                       Define "annodof3" for Elem. No. annotation in Plot 11.
.                       Define "annodof4" for Node No. annotation in Plot 11.
.                       Define "annodof5" for dof no. annotation in Plot 11.
.
. ---------------------------------------------------------------------
.
.     References:
.
.     [1978]  Box, G. E., Hunter, W. G., and Hunter, J. S.,
.             "Statistics for Experimenters: An Introduction to Design
.             Data Analysis, and Model Building."  Wiley (1978).
.
.     [2002]  Filliben, J. J., and Heckert, N. A., "DATAPLOT:  A
.             Statistical Data Analysis Software System," A Public-
.             Domain Software Released by NIST, Gaithersburg, MD 20899.
.             http://www.itl.nist.gov/div898/software/dataplot.html.
.
.     [2003]  Croarkin, C., et al, eds., "NIST/SEMATECH e-Handbook of
.             Statistical Methods, Chapter 5 on Process Improvement."
.             http://www.itl.nist.gov/div898/handbook/, first issued
.             June 1, 2003, and last updated July 18, 2006.  Produced
.             jointly by NIST Statistical Engineering Division and the
.             Statistical Methods Group of SEMITECH.  Also available
.             as a NIST Interagency Report in a CD-ROM upon request to
.             alan.heckert@nist.gov.
.
.     [2008]  Fong, J. T., Filliben, J. J., Heckert, N. A., and deWit,
.             R., "Design of Experiments Approach to Verification and
.             Uncertainty Estimation of Simulations Based on Finite
.             Element Method," Proceedings of Annual Conference of the
.             American Society for Engineering Education (ASEE), June
.             22-25, 2008, Chicago, IL, Paper No. AC2008-2725 (2008).
.
.     [2009]  Fong, J. T., deWit, R., Marcal, P. V., Filliben, J. J.,
.             and N. A. Heckert, "A Design-of-Experiments Plug-In for
.             Estimating Uncertainties in Finite Element Simulations,"
.             Draft Manuscript dated Feb. 6, 2009 as submitted to the
.             SIMULIA Customer Conference (May 18-21, 2009, London UK)
.             for review and possible publication as a conf. paper.
.
. ---------------------------------------------------------------------
.
dimension 100 columns
set fatal error prompt
.
```

```
. ===== Plot Note-1: User to define a plot color variable named "plcolor"
.
let string plcolor = black
.
. ================================================= End of Plot Note-1
device 2 color on
set postscript font helvetica bold
let string foreground = ^plcolor
let string background = white
.
if foreground not exist; let string foreground = white; end if
if background not exist; let string background = ^plcolor; end if
.
background color ^background
.
char color ^foreground all
line color ^foreground all
bar border color ^foreground all
spike color ^foreground all
frame color ^foreground
tic color ^foreground
tic label color ^foreground
label color ^foreground
title color ^foreground
legend color ^foreground
color ^foreground
.
. -----end color specification-----
.
let string trailer = pedro7.dp
hw 2.4 1.2; just right; move 98 94; text ^trailer
.
. ========== Plot Note-2:   currdate varies with application ==========
.
let string currdate = 2/26/09 3:00 EST
.
. ================================================= End of Plot Note-2
.
. ------------ read output of button_1 (Python code PyDpGui_1.py)
skip 0
row limit 3 3
read parameter DpGuiOut_1.txt k
.
let kmax1 = k*2 + 6
let kmax2 = k*2 + 7
let kmin1 = kmax1 - k*2 + 2
let kmin2 = kmax2 - k*2 + 2
.
let icnt = 0
Loop for kk = kmin1 2 kmax1
  row limit kk kk
  let icnt = icnt + 1
  read string DpGuiOut_1.txt TZ^icnt
end of loop
.
let icnt = 0
Loop for kk = kmin2 2 kmax2
  row limit kk kk
```

```
  let icnt = icnt + 1
  read string DpGuiOut_1.txt SYMB^icnt
end of loop
.
Let rowz = k*2 + 9
row limit rowz rowz
serial read DpGuiOut_1.txt z
.
let rowp = k*2 + 11
row limit rowp rowp
serial read DpGuiOut_1.txt pct
.
. ---------------- read output of button_2 (Python code PyDpGui_2.py)
.
let string filename = DpGuiOut_2.txt
.
row limit 7 7
read parameter ^filename n
.
row limit 12 12
read string ^filename TEXTY
row limit 13 13
read string ^filename SYMBY
row limits
skip 14
serial read ^filename w
.
. --------------- read a (k,n) design from a subdirectory named
.                 doe_tables using Python code, PyDpGui_2.py,
.                 that creates an ASCII file named pedro7.txt
row limits
skip 13
read pedro7.txt x1 to x^k
skip 0
stat v
.
. ---------------------------- assign center point w(1)
Let ydatacen = w(1)
.
. ---------------------------- re-order w(2) through w(n+1)
let y1 = w
let ntemp = size y1
retain y1 for i = 2 1 ntemp
let y = y1
.
. ======== Plot Note-3: strings "project", "finelem", "xxlabel" to vary
.                      with application.  To be changed by user.
.
let string finelem = Theoretical Solution
let string project = Isotropic Silicon Cantilever Beam Free Vibration ( ^finelem )
let string xxlabel = First Bending Resonance Frequency  q1  ( kHz )
.
. ================================================== End of Plot Note-3
.
let string cy = ^TEXTY
let icnt = 0
Loop for icnt = 1 1 k
let string cx^icnt = ^SYMB^icnt
```

```
end of loop
.
. ------------ Start 10-step analysis and generate 10 DOE plots
.
skip 0
row limits
column limits
.
call dexplot.dp
.
. ---------------------------------------------------------------------
.
.  Postscript:
.  Analysis Step 10 (Contour Plot) did put the two dominant factors
.  in two new variables named U1 and U2.
.
.  Contour plot does not have center point information, but variables
.  U1 and U2 will contain center point if furnished by user.
.
.  Step 11 is added to deliver Plot 11 (95% confidence bounds plot)
.
. ---------------------------------------------------------------------
.
.     Step 11:   if centerpoint exists, append it prior to the fit
.                if center point not exist, then do nothing pre-fit
.
. ---------------------------------------------------------------------
let n = number y1
if ydatacen exists
   let np1 = n + 1
   let y1(np1) = ydatacen
   let u1(np1) = 0
   let u2(np1) = 0
end if
.
FIT y1 U1 U2
SKIP 0
READ PARAMETER DPST1F.DAT A0 SDA0
.
LET SDY2 = SDA0**2 + (A1**2/4) + (A2**2/4)
LET SDY  = SQRT(SDY2)
.
LET TVAL = TPPF(0.025,RESDF)
Let TVAN = TVAL*SQRT(1+1/n)
.
LET YLCL = A0 + SDY*TVAN
LET YUCL = A0 - SDY*TVAN
let halfint = a0 - ylcl
.
let a0 = round(a0,3)
let ylcl = round(ylcl,3)
let yucl = round(yucl,3)
let halfint = round(halfint,3)
echo off
.
CAPTURE pedro7_UNC1.TXT
printing on
FIT y1 U1 U2
```

```
let a0 = round(a0,3)
printing off
WRITE " "
WRITE " "
WRITE "          Uncertainty Analysis"
WRITE " "
WRITE "A0                                = ^A0"
WRITE "A1                                = ^A1"
WRITE "A2                                = ^A2"
WRITE "Resdiual SD                       = ^RESSD"
WRITE "Resdiual DF                       = ^RESDF"
WRITE "Variance(Y)                       = ^SDY2"
WRITE "SD(Y)                             = ^SDY"
WRITE "Lower 95% Confidence Bound for Y  = ^YLCL"
WRITE "Upper 95% Confidence Bound for Y  = ^YUCL"
WRITE " "
END OF CAPTURE
.
. ---------------------------------------------------------------------
.
.     The above Dataplot code created an estimated mean, A0,
.     and an estimated standard deviation, SDY, for a 2-variable
.     data set of sanmple size,  n+1, if a center point exists.
.
.     A normal plot of the results of the k-factor, (n+1)-run, 10-step
.     analysis, and 2-variable least square fit for estimating the 95%
.     uncertainty upper and lower bounds have been made with the
.     Step-11 Dataplot code.
.
. ---------------------------------------------------------------------
let m1 = A0
.
. ----- define left x-limit as xLL, rounded to xLLr
let xLL = m1*0.85
let xLLr = round(xLL, 0)
.
. ----- define right x-limit as xLR, rounded to xLRr
let xLR = m1*1.15
let xLRr = round(xLR, 0)
.
xlimits  xLLr  xLRr
ylimits  0  0.15
x3label off
let sleft1 = 0.95*YLCL
let sright1 = 1.05*YUCL
.
line solid
line thickness 0.45
title offset 2
title case asis
label case asis
tic mark label case asis
x1label size 4
x1label ^xxlabel
xtic label size 3
.
y1label size 5
y1label displacement 10
```

```
y1label Probability
y1tic label size 4
plot norpdf(x,m1,SDY) for x = sleft1 .001 sright1
.
line thickness 0.35
lines dash; drawdddd m1 0 m1 0.115; lines blank
line thickness 0.3
lines dotted; drawdddd YLCL 0 YLCL 0.045; lines blank
lines dotted; drawdddd YUCL 0 YUCL 0.045; lines blank
move 17 58; just lef; hw 3.6 1.8; text 95% Lower
move 17 53; just left; hw 3.6 1.8; text     Bound =
move 70 50; just left; hw 3.6 1.8; text 95% Upper
move 72 45; just left; hw 3.6 1.8; text   Bound =
let m2 = 1.03*m1
lines dotted; drawdddd m1 0.110 m2 0.116; lines blank
let m4 = YUCL
let m5 = 1.05*m4
lines dotted; drawdddd m4 0.02 m5 0.03; lines blank
let m6 = YLCL
let m7 = 0.98*m6
lines dotted; drawdddd m7 0.052 m6 0.033; lines blank
.
. ========= Plot Note-4:  User to specify below extra annotations in Plot 11
.
let string bxlabel = Beam L = 0.232 mm, h = 0.007 mm, E (Young's Modulus) =
169,158.0 MPa.
let string annodof1 = Solution Type:  Theoretical (Exact)
let string annodof2 = Ref.: Timoshenko & Young, 1955
let string annodof3 = rho = 2.329 e-9 tons/cu mm
let string annodof4 = q1 = (1/(2pi))*(1.875)**2 *
let string annodof5 = sqrt(E/(12rho))*(h/(L*L))
.
. ========================================================= End of Plot Note-4
.
move 50 96.5; just center; hw 3.6 1.8; text ^project
move 50 92; just center; hw 3.5 1.75; text Python-Dataplot Uncertainty Plug-In,
PD-UP_v.1.0  ( Fong, et al, 2009 )
move 50 3; just center; hw 2.6 1.3; text ^bxlabel
.
move 17 86; just left; hw 3 1.5; text ^annodof1
move 17 82; just left; hw 3 1.5; text ^annodof2
move 17 78; just left; hw 3 1.5; text ^annodof3
move 17 74; just left; hw 3 1.5; text ^annodof4
move 21 70; just left; hw 3 1.5; text ^annodof5
move 60 71; just left; hw 4.8 2.4; text ^a0 ( ^halfint ) kHz
justification right
move 94 65; hw 3.2 1.6; text ( for a  ^np1 - run "DOE" with k = ^k,
move 94 60; hw 3.2 1.6; text n = ^n  +  center point  =  ^np1 )
move 99 9.5; hw 2 1; text ^currdate
move 99 6; hw 2.4 1.2; text ^trailer
move 17 47; just left; hw 4.4 2.2; text ^ylcl  kHz
move 72 39; just left; hw 4.4 2.2; text ^yucl  kHz
.
limits
erase
quit
. ----------- END OF DATAPLOT PROGRAM  pedro7_a90226_k3_n8+1_Exact.dp -----------
```