Design of an intelligent PYTHON code to run coupled and license-free finite-element and statistical analysis software for calibration of near-field scanning microwave microscopes[1]

Jeffrey T. Fong[2*], N. Alan Heckert[3], James J. Filliben[3], Pedro V. Marcal[4], Samuel Berweger[5], T. Mitchell Wallis[5], Kristen Genter[5], and Pavel Kabos[5]
1. Contribution of the National Institute of Standards & Technology (NIST). Not subject to copyright.
2. Applied & Computational Mathematics Division, NIST, Gaithersburg, MD 20899-8910, U.S.A.
*Corresponding author contact, *fong@nist.gov*, or, *fong70777@gmail.com*
3. Statistical Engineering Division, NIST, Gaithersburg, MD 20899-8960, U.S.A.
4. MPact Corp., Oak Park, CA 91377, U.S.A.
5. Applied Physics Division, NIST, Boulder, CO 80301, U.S.A.

Abstract: To calibrate near-field scanning microwave microscopes (NSMM) for defect detection and characterization in semiconductors, it is common to develop a parametric finite element analysis (FEA) code to guide the microscope user on how to optimize the settings of the instrument to improve its performance. Two problems arise that make the application of the FEA code difficult if not impossible. The first problem is due to the approximate nature of the FEA method and the critical requirement that the accuracy of the FEA solutions be mathematically verified during the entire calibration process. The second problem is a pre-requisite that the user's computer be licensed with the specific FEA software at a sizable cost and training time to the user. In this paper, we solve both problems by designing an intelligent PYTHON code that manages the seamless running of two license-free codes, namely, a compiled parametric COMSOL AC/DC-Module-based code that yields a series of solutions at various finite element mesh densities as input to a FEA-verification code written in a statistical analysis software named DATAPLOT that uses a nonlinear least squares method to check and verify the FEA solution of the COMSOL code. An example of a generic NSMM calibration code running a coupled and license-free finite element and statistical analysis software is presented and discussed.

Keywords: Computational modeling, COMSOL, DATAPLOT, electromagnetics, element type, FEM, finite element method, logistic function, mesh density, near-field scanning microwave microscopy, nonlinear least squares method, PYTHON, statistical analysis, uncertainty quantification.

Disclaimer: Certain commercial equipment, materials, or software are identified in this paper in order to specify the computational procedure adequately. Such identification is not intended to imply endorsement by NIST, nor to imply that the equipment, materials, or software identified are necessarily the best available for the purpose.

1. Introduction

To calibrate near-field scanning microwave microscopes (NSMM), as shown schematically in Fig. 1, for detection, sizing, and depth estimation of surface and subsurface defects in semiconductors, it is common to develop a parametric finite element

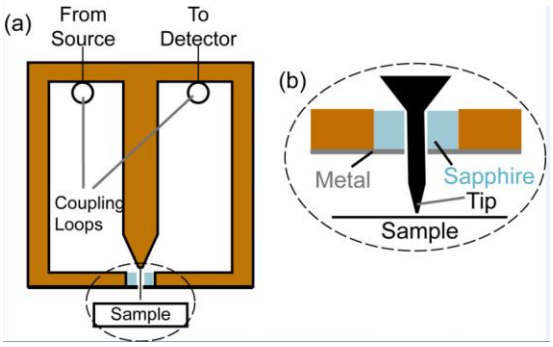

Figure 1. A near-field scanning microwave microscope (NSMM) design based on a coaxial resonator. (a) A schematic of the coaxial resonator, which is operated in transmission mode. (b) A closer view of the prober tip, which is connected to the center conductor of the resonator and extends toward the sample through an opening at the bottom of the resonator (after Gao and Xiang [2] as reproduced in Fig. 7.6, p.115 of a book by Wallis and Kabos [1]).

analysis (FEA) code to guide the microscope user on how to optimize the settings of the instrument to improve its performance.

Two problems arise that make the application of the FEA methodology difficult if not impossible. The first problem is due to the approximate nature of the FEA method and the critical requirement that the accuracy of the FEA solutions be mathematically verified during the entire calibration process (see, e.g., refs. [3] through [7]). The second problem is a pre-requisite that the user's computer be licensed with a specific FEA software capable of solving the calibration problem (such as COMSOL [8]) at a sizable cost and training time to the user.

In this paper, we solve both problems by formulating the conceptual design of an intelligent PYTHON code that manages, as shown by Chollet [9] and Fong, et al [10], the seamless running of two license-free[6] codes, namely, a compiled [11] parametric COMSOL AC/DC-Module-based code that yields a series of solutions at various finite element mesh densities as input to a FEA-verification code written in a statistical analysis software named DATAPLOT [12] that uses a nonlinear least squares logistic function fit method [6, 7] to verify the FEA solution of the COMSOL code.

To present the design concept of an intelligent PYTHON code for NSMM calibration, we choose to use an example problem with a very simple sample geometry, namely, a small spherical inclusion in a semiconductor block of 2 um length, 1 um width, and 0.5 um thickness. The diameter of the inclusion is 25 nm, and the clear distance from the top of the sphere to the top surface of the block is 33.3 nm.

In addition, we prescribe that the electrode radius is 10 nm, the electrode voltage is 0.1 v, and the operating frequency is 5 GHz.

To simplify the problem further, we also prescribe the material properties of the block and the inclusion without an estimate of uncertainty, namely,

The block electric conductivity = 1e-12 S/m.

The block dielectric constant = 11.7.

The inclusion electric conductivity = 0.01 S/m.

The inclusion relative permittivity = 20.

In Section 2, we develop a parametric finite element analysis (FEA) code using the AC/DC module of the COMSOL code [8] and the COMSOL Compiler code [11] to obtain an executable code that runs in any Windows-based laptop without a COMSOL license.

In Section 3, we develop a FEA verification code written in DATAPLOT [12] to check whether the finite element solutions at increasing mesh densities converge to an asymptotic solution with an acceptable measure of uncertainty. In Section 4, we design a PYTHON code to manage the running of a coupled FEA code and a verification code such that a verified FEA solution is achieved to a prescribed level of uncertainty as required by the calibration mission.

[6]COMSOL's license terms state that a code generated by the COMSOL Compiler may be executed by anyone without the need to access a license file. Thus, the generated program can be run by anyone, including those who do not have a paid COMSOL subscription. In this paper we use the term "license-free" when describing these particular terms-of-use for a COMSOL Compiler-generated code**.** It does not apply to DATAPLOT, a non-commercial code.

2. Finite Element Analysis Code for a Simple NSMM Flaw Depth/Size Estimation Problem

In Figs. 2 and 3, we show the governing equations and a list of 19 adjustable parameters for the COMSOL FEA code of the simple NSMM problem, respectively. In Figs. 4, 5, and 6, we show

$$\nabla \cdot \mathbf{J} = Q_j$$
$$\mathbf{J} = \sigma \mathbf{E} + j\omega \mathbf{D} + \mathbf{J}_e$$
$$\mathbf{E} = -\nabla V$$

Note: **J** is the current density, Q_j, the current source, σ, the conductivity, **E**, the electric field, ω, the frequency, **D**, the electric flux density, $\mathbf{J}_e$, the external current density, V, the electric potential, and $j = \sqrt{-1}$.

Figure 2. Governing equations of the example problem as shown in the COMSOL AC/DC Module [8] screen output.

block, x-dimension:	2000	nm
block, y-dimension:	1000	nm
block, z-dimension:	500	nm
diameter of inclusion:	5[nm]*5	m
distance from top:	100[nm]/3	m
x-location of inclusion:	0	nm
electrode radius:	1[nm]*10	m
electrode offset:	500	nm
electrode voltage:	0.1	V
operating frequency:	5	GHz
chip elec. conductivity:	1e-12	S/m
chip dielectric constant:	11.7	
inclusion electrical conductivity:	0.01	S/m
inclusion relative permittivity:	20	
no. of elements per inclusion diameter:	5	
no. of elements per electrode radius:	5	
add elements to electrode:	3	
max. element size_in_chip:	200	nm
ratio of refinement of max. mesh size in chip:	0.50	

Figure 3. A list of parameters in the COMSOL FEA code that are designed to be changed by a user as needed.

the typical sample geometry and mesh design. After we completed the COMSOL code, we compiled it to get an executable code that runs in a license-free laptop (see Fig. 7 for a screen output display of the executable code). Typical results of either the original or the compiled code appear in Figs. 8 and 9.

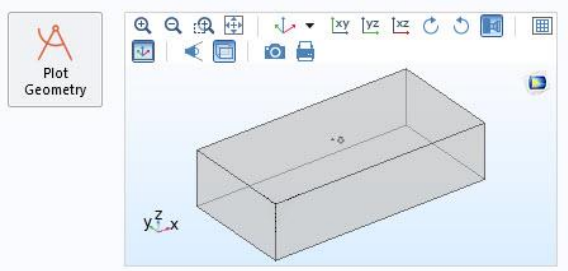

Figure 4. A screen display of the semiconductor block with a small spherical inclusion buried below the top.

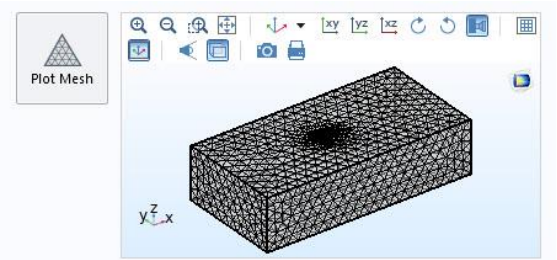

Figure 5. A typical finite element mesh design for the semiconductor block with a buried spherical inclusion where all the elements are tetrahedrons. It is worth noting that meshing in finite element analysis of electromagnetism problems differs from that of structural problems in one important feature, namely, the degrees of freedom of electromagnetic elements are associated with edges, faces, cells, not nodes. For brevity, we shall make no attempt in this paper to explain in details how the meshing was done in a COMSOL code except by referring to its manual [8].

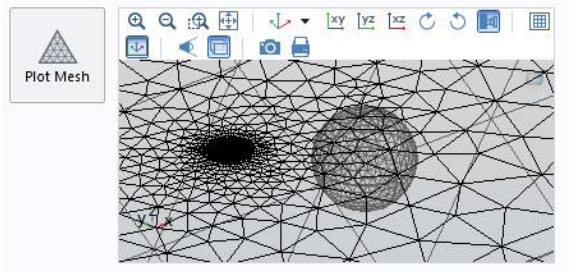

Figure 6. An enlarged view of the finite element mesh design for the buried spherical inclusion with a simultaneous display of the surrounding mesh without the presence of the inclusion. For details, see manual [8].

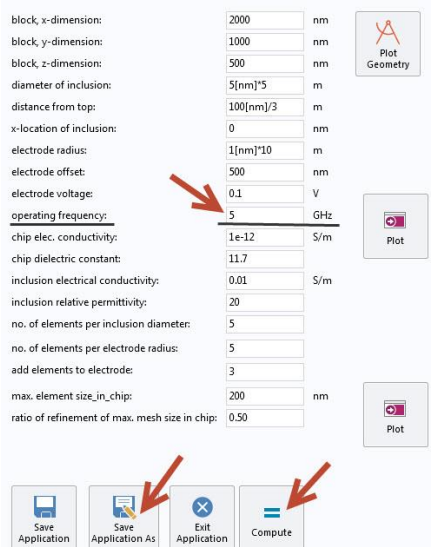

Figure 7. Screen output display of running a compiled COMSOL FEA code after the original COMSOL code is compiled. Note the presence of all adjustable parameters and buttons to re-compute and to save results of a new run.

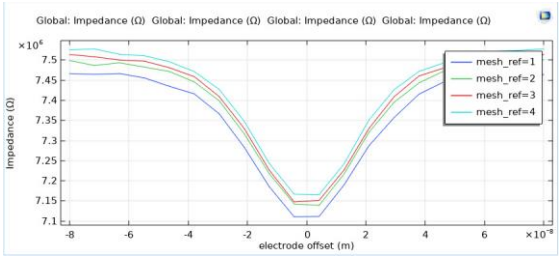

Figure 8. Impedance results of the first four runs at four different finite element mesh densities to check convergence of solutions.

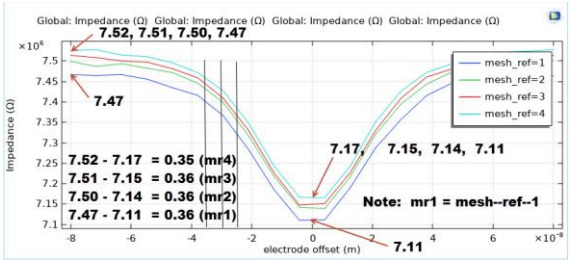

Figure 9. Readings for a key result of the FEA runs for four different mesh densities. When the probe is located directly over the center of the spherical inclusion, the values of the impedance correspond to four mesh densities are:

Mesh Ref. No.	d.o.f.	Impedance (million ohms)
1 (coarser)	51,020	7.11
2	52,058	7.14
3	53,537	7.15
4 (finer)	55,313	7.17

3. A Finite Element Solution Verification Code

To obtain an asymptotic solution of a sequence of FEA candidate solutions at increasing mesh densities (or degrees of freedom, d.o.f.), we apply a 4-parameter nonlinear least-squares (NLLSQ) logistic function fit method (see, e.g., Fong, et al. [6, 7]). A minimum of five candidate solutions is required to run the fit method. For the results given in Fig. 9, we had only four candidate solutions, so we need to run the executable FEA code one more time to generate a fifth candidate solution as follows:

Mesh Ref. No.	d.o.f.	Impedance (million ohms)
1 (coarser)	51,020	7.11
2	52,058	7.14
3	53,537	7.15
4 (finer)	55,313	7.17
5 (new)	**57,089**	**7.18**

In Fig. 10, we show a plot of the NLLSQ fit of 5 points with an asymptotic solution = 7.186 M-ohms. In Fig. 11, we re-plot the result of the 5-point fit with an estimate of uncertainty defined at one million d.o.f. For the calibration purpose in mind, that uncertainty (= 14 %) is too large, so we need to add more points until the uncertainty is less than 2 %. In Figs. 12-14, we show that the goal is reached where the number of candidate solutions reaches nine.

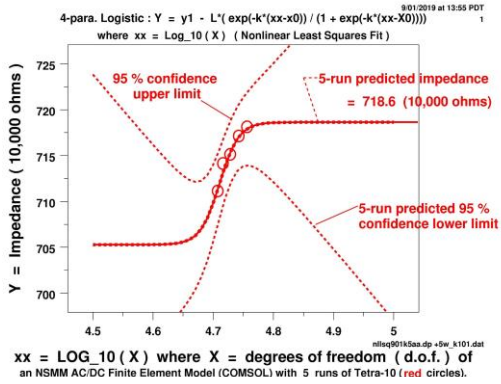

Figure 10. A NLLSQ logistic function fit of 5 candidate solutions yields an asymptotic solution = 7.186 M-ohms**.**

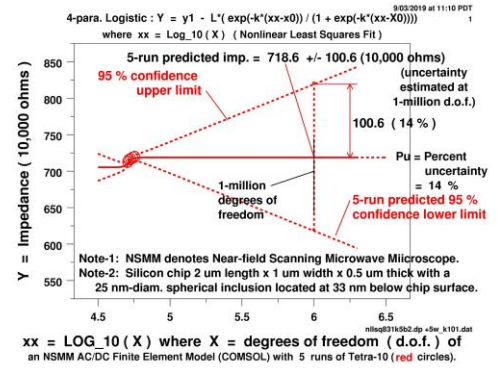

Figure 11. A re-plot of the NLLSQ fit of the 5 candidate solutions with 95 % confidence limits and an estimate of the uncertainty as defined for a million d.o.f. The asymptotic solution now equals 7.186 +/- 1.006 M-ohms.

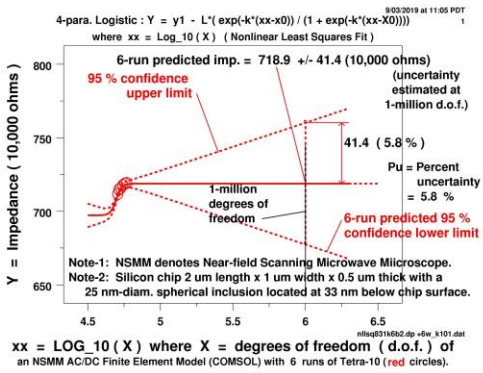

Figure 12. A NLLSQ fit of 6 candidate solutions yields an asymptotic solution equal to 7.189 +/- 0.414 M-ohms.

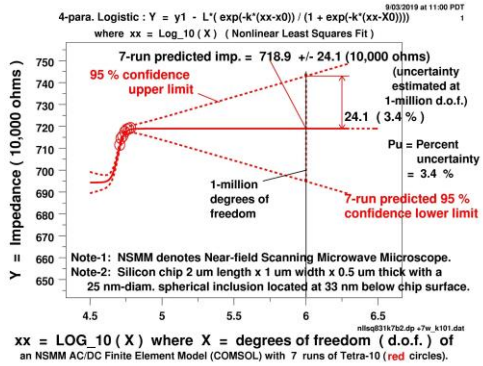

Figure 13. A NLLSQ fit of 7 candidate solutions yields an asymptotic solution equal to 7.189 +/- 0.241 M-ohms.

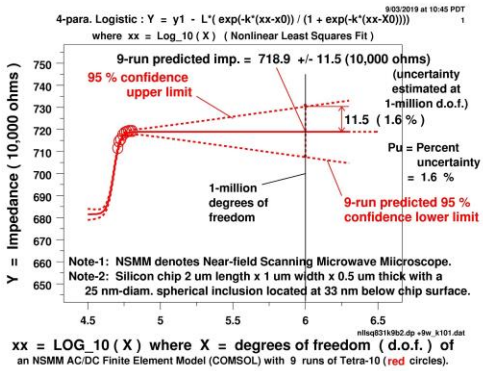

Figure 14. A NLLSQ fit of 9 candidate solutions yields an asymptotic solution equal to 7.189 +/- 0.115 M-ohms.

4. Design of a PYTHON Calibration Code

As shown by Chollet [9], PYTHON is highly suited as a language for writing AI codes in general, because PYTHON code acts as a manager to call on all types of application codes to run on different platforms either sequentially, iteratively, or both.

For example, the following chunk of codes, written in PYTHON, is part of a larger code that will allow a user to run a COMSOL application code named "Fong_Kabos_app_Altasim_paid_license.exe":

```
Def InverseProgram() :
  self_m_fileInput=True
  print '***  Run ComsolManager.py '
  total_volume=0.0
  if self_m_fileInput :
    returncode=None
    try :
      z=y[0]+'.dat'
      comsol_data=os.path.join(dir_name,z)
      #returncode= call(['C:/
Fong_Kabos_app_Altasim_paid_license.exe' ,
comsol_data])
      returncode= subprocess.Popen(['C:/
Fong_Kabos_app_Altasim_paid_license.exe' ,
comsol_data])
      #subprocess.Popen.terminate()
      pass
      if returncode :
        print 'Failure with return code' ,returncode
    except :
      pass
InverseProgram()
```

In short, a PYTHON code can be written to run two application codes, namely, a parametric FEA code to solve an NSMM problem and a FEA verification code to obtain an asymptotic solution within a prescribed level of uncertainty. Since it is desirable for all NSMM calibration codes to run on Windows-based computers without FEA software licenses, we show in this paper that with a compiled COMSOL finite element code and an open-source DATAPLOT statistical analysis code, it is feasible to design an intelligent PYTHON code to calibrate instruments such as the near-field scanning microwave microscope.

5. A License-free Calibration Code

The availability of a compiler for COMSOL FEA codes makes it possible for engineers to design all kinds of parametric FEA codes that will run in any Windows laptop without a FEA software license. In Fig. 15, we show how to change the operating frequency from 5 GHz to 2 GHz and re-compute to get new results in a laptop without a software license. The new results are given in Fig. 16. Using the PYTHON code, we can obtain a FEA asymptotic solution to any prescribed degree of uncertainty.

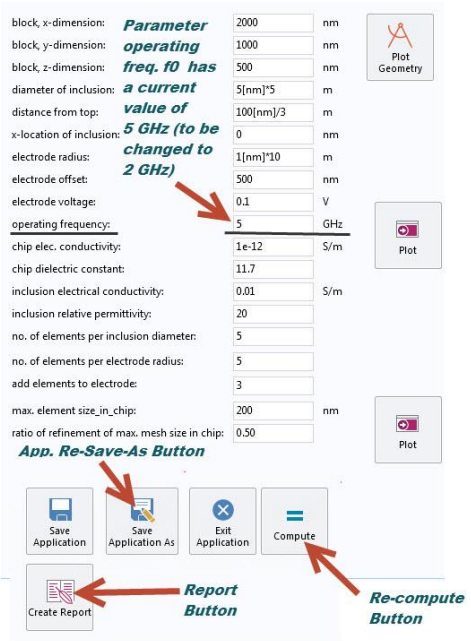

Figure 15. A screen display of the compiled COMSOL code with 19 adjustable parameters and buttons to re-compute, save new application, and create report.

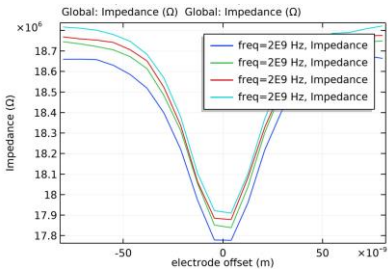

Figure 16. Impedance results for the simple NSMM spherical inclusion problem where the operating frequency has just been altered from 5 GHz to 2 GHz. Since the run only produced 4 candidate solutions, and the NLLSQ fit method requires a minimum of 5 candidates, the intelligent PYTHON code will automatically activate the executable COMSOL code to produce a 5th point.

6. Significance and Limitations of the Conceptual Design of an Intelligent Calibration Code

The conceptual design of an intelligent PYTHON code to run license-free coupled FEA and verification codes is significant in at least two innovative aspects of engineering, namely, (a) it removes a dependency on FEA license for heavy use of candidate runs to achieve solution accuracy, and (b) it provides the engineer access to a sophisticated FEA solution verification tool that would otherwise be unused for lack of lengthy training. It is obvious that the idea of designing an intelligent calibration code for NSMM is not limited only to a specimen with a spherical inclusion, because the finite element method is quite general. However, the proposed design is not without limitations, chiefly among which is the limited emphasis on seeking an asymptotic solution based on mesh density variations. It ignores uncertainty due to other factors such as material properties, voltage and frequency accuracy.

7. Concluding Remarks

Finite element analysis (FEA) methodology is a powerful tool to assist engineers and scientists in calibrating precision instruments such as near-field scanning microwave microscopes (NSMM). The availability of a compiler for any FEA software to convert the source code into an executable code that runs in a laptop without a software license is a major step forward to broaden the usage of FEA methodology. The introduction of an intelligent PYTHON code to run seamlessly both the FEA executable code and a FEA verification code in order to achieve a verified FEA asymptotic solution to a prescribed degree of uncertainty is a much-needed addition to the calibration group of the scientific measurement community.

8. References

1. Wallis, T. M., and Kabos, P., *Measurement Techniques for Radio Frequency Nanoelectronics*. Cambridge University Press (2017).
2. Gao, C., and Xiang, X. -D., "Quantitative Microwave Near-Field Microscopy of Dielectric Properties," *Review of Scientific Instruments 69, pp. 3846-3851* (1998).
3. Zienkiewicz, O. C., *The Finite Element Method in Engineering Science, 3rd ed., pp. 190-191*. McGraw-Hill (1977).
4. Zienkiewicz, O. C., and Taylor, R. L., *The Finite Element Method, 5th ed., Vol. 1: "The Basis," Sections 8.3 and 8.4, pp. 168-172*. Butterworth-Heinemann (2000).
5. Jin, J., *The Finite Element Method in Electro-magnetics, 2nd ed.* Wiley (2002).
6. Fong, J. T., Heckert, N. A., Filliben, J. J., Marcal, P. V., and Rainsberger, R., "Uncertainty of FEM Solutions Using a Nonlinear Least Squares Fit Method and a Design of Experiments Approach," *Proc COMSOL Users' Conf, Oct. 7-9, 2015, Boston, MA,www.comsol.com/ed/direct/conf/conference2015 papers/papers/* (2015).
7. Fong, J. T., Filliben, J. J., Heckert, N. A., Marcal, P. V., Rainsberger, R., and Ma, L. "Uncertainty Quantification of Stresses in a Cracked Pipe Elbow Weldment Using a Logistic Function Fit, a Nonlinear Least Squares Algorithm, and a Super-parametric Method," *Procedia Engineering, 130, pp. 135-149* (2015).
8. COMSOL, *AC/DC Module User's Guide, Version 5.4, www.comsol.com* (2019).
9. Chollet, F., 2017, *Deep Learning with Python*. Manning Publications (2017).
10. Fong, J. T., Marcal, P. V., Rainsberger, R., Heckert, N. A., and Filliben, J. J., "Design of an intelligent PYTHON code for validating crack growth exponent by monitoring a crack of zig-zag shape in a cracked pipe," *Proc. ASME Pressure Vessels and Piping Division Conf., July 14-19, 2019, San Antonio, TX, U.S.A., Paper PVP2019-93502*. New York, NY: Amer. Soc. Mech. Engineers (2019).
11. COMSOL, *Compiler Module User's Guide, Version 5.4, www.comsol.com* (2019).
12. Filliben, J. J., and Heckert, N. A., 2002, Dataplot: A Statistical Data Analysis Software System, National Institute of Standards & Technology, Gaithersburg, MD 20899, http://www.itl.nist.gov/div898/software/dataplot.html , 2002.

9. Acknowledgment

We wish to thank Drs. **Sergei Yushanov**, **Kyle Koppenhoefer**, **Joshua Thomas**, and **Jeffrey Crompton**, all of Altasim Technologies, LLC, Columbus, OH, for their technical assistance during this investigation. In addition, we wish to thank Altasim Technologies, LLC, for providing the first author of this paper (J. T. Fong) access to a licensed copy of the COMSOL Compiler module for obtaining new research result regarding the conversion of a COMSOL source code (extension .mph) to a compiled application code (extension .exe).