2025 IMEKO TC-6 International Conference on
Metrology and Digital Transformation - M4DConf 2025
Benevento, Italy, September 3-5, 2025

A Python-based graphical uncertainty calculator with optimal propagation of uncertainty and Monte-Carlo evaluation possibility

Khaled M. Ahmed[a,*], *Ali Q. Alanbari*[a] *and Abdullah S. Alosaimi*[a]

[a] National Measurement and Calibration Center, Saudi Standards, Metrology and Quality Org. (SASO-NMCC), Riyadh, Saudi Arabia,
*Corresponding author: k.abdelftah@saso.gov.sa / khaled55eg@gmail.com

Abstract **– A web-based uncertainty calculator for metrology professionals is presented in this paper. The system incorporates a Monte Carlo (MC) engine in accordance with GUM Supplement 1 and optimal first-order propagation that complies with the Guide to the Expression of Uncertainty in Measurement (GUM). A case study, algorithms, and architecture are described in detail. The findings demonstrate enhanced transparency, clear report generation (in PNG/PDF format) suitable for ISO/IEC 17025 documentation, and strong agreement between analytical propagation and MC evaluation. The benefits of integrating statistical simulation and rigorous analytical propagation in an intuitive web-based platform are emphasized. Users can view the installed script file and make changes to continuously improve its functionality using the editor-like text window that outputs the calculation results.**

Keywords**:** Digital Metrology, Uncertainty Calculator, Python-based algorithms, GUM, Monte Carlo Simulation.

1. MOTIVATION

When both the measured quantity and its uncertainty are provided, all measurements can be considered complete. Given the significance of this uncertainty, it makes sense that standards for expressing uncertainty have been developed and are widely applied. For general metrologists, evaluating uncertainty in accordance with these expression guidelines is not an easy task. When the number of independent variables corresponding to each uncertainty component increases or the model equation becomes complex, it becomes necessary to perform laborious partial differentiation in order to calculate the sensitivity coefficient for each component, even if the measurement uncertainty model equation can be identified and each uncertainty component can be estimated. If the probability distribution of the propagated uncertainty does not follow a normal distribution, the coverage factor corresponding to a certain confidence interval (e.g., 95%) is not simply twice the composite standard uncertainty ($k = 2$) [1-6]. Two program libraries for simply solving this mathematical problem have been released in the Python language. However, setting up and using the language environment is not an easy task for users who are unfamiliar with this language. This serves as a barrier to learning Python. The development of open source software is free because many helpful open libraries for Python are freely available. Python is a powerful, flexible, and accessible programming language that offers numerous advantages. It is easy to learn, open-source, and free, running on various platforms with minimal changes. It has numerous libraries for scientific computing, data analysis, and machine learning. Python's high-level nature allows quick prototyping and faster coding. It also has integration capabilities with other languages and tools. Its strong community and wide user base make problem-solving easier. Python is versatile, used in web development, automation, AI, and more. It supports object-oriented and functional programming and is ideal for automation and scripting.

Python is a widely used and open-source uncertainty calculator that offers several benefits. It is accessible to students, labs, and institutions without high software costs. Python automates complex calculations, reducing human error. Its scripts can be saved, versioned, and shared, ensuring traceability. Users can customize the code for specific models, such as GUM and Monte Carlo methods. Python also provides educational value by helping learners understand uncertainty analysis logic. It integrates easily with Excel, databases, and instruments, linking uncertainty analysis with real-time measurements or calibration workflows. Python-based tools contribute to digital transformation, aligning with metrology 4.0 initiatives [7-11].

Measurement science necessitates frequent statistical processing and numerical computations, just like other scientific domains. The uncertainty calculation module is only one example of using the Python engine built into this demo program. If scripts for other purposes are developed and shared in the future, this engine can function as a shared platform for calculations as its usability increases.

An essential component of metrology and scientific research is the assessment of measurement uncertainty. In order to make it easier to propagate measurement uncertainties through user-defined mathematical models, this paper presents uncertainty calculator, a lightweight, open-source Python application. This calculator provides an approachable platform for both instructional and real-world applications in uncertainty analysis by utilizing the SymPy library for symbolic mathematics and Tkinter/JSON for the graphical user interface. Through case studies, the tool's capabilities are illustrated, emphasizing both its potential to advance digital transition in metrology and its usefulness in calibration operations [9-14].

To address uncertainty evaluation in calibration and testing, a calculator built with Python was created. As presented case study, the calculator ensures optical metrology realism and auditable outputs by combining a high-throughput Monte Carlo engine with GUM-compliant propagation. The calculator features domain-specific models, an online interface, and automatic PNG/PDF report generation. It has been improved to have a simple web user interface and MC capability. Real-world example includes optical metrology is presented. There is also comparison report that summarizes the methodology, benefits, and outcomes.

2. PREVIOUS WORK

A thorough list of the top software tools and uncertainty calculators for metrology, arranged by kind, features, common applications, and real-world restrictions. NPL software for GUM Supplement 1 examples, NMI-developed libraries and frameworks, and NMI-built web apps are the three primary categories into which the tools fall. NPL software for GUM Supplement 1 examples is a collection of Windows programs that replicate GUM S1 example problems, while NIST Uncertainty Machine (UM) is a free web-based tool that applies the Gauss formula and Monte Carlo (GUM S1) law of propagation. The sophisticated library METAS UncLib/metas-unclib offers linear, higher-order, numerical, and Monte Carlo propagation for multivariate, correlated, and complex-valued quantities. While Uncertainties uses first-order Taylor series to automate uncertainty propagation, MetroloPy is a Python toolkit for GUM-consistent computations. MUkit is a training ecosystem and software for chemical and environmental uncertainty that complies with ISO 11352 for water quality and Nordtest TR 537. GUM Workbench, GUMsim, and QMSys GU M are commercial desktop workbenches that are best suited for particular uses. The article highlights new developments in digital metrology, automated experiments for uncertainty, and the development of multivariate/MC capabilities while offering helpful selection advice for labs utilizing GUM and MC in their analysis tools [13, 15].

3. PROGRAM FEATURES AND CORE FUNCTIONALITIES

In order to ensure confidence in results and compliance with ISO/IEC 17025 and the Guide to the Expression of Uncertainty in Measurement (GUM), measurement uncertainty evaluation is essential in modern metrology. This Python-based graphical uncertainty calculator offers an adaptable, transparent, and user-friendly solution, in contrast to traditional calculators that frequently provide simple analytical propagation or restricted Monte Carlo capabilities. The calculator integrates analytical (GUM) uncertainty propagation using sensitivity coefficients, Monte Carlo simulation for non-linear models or complex uncertainty propagation, and an interactive, professional GUI for input, results, and report generation. The system is a multidisciplinary tool for labs and R&D facilities since it supports general measurement models in various fields. Its primary features include Monte Carlo simulation, the Analytical Propagation (GUM Method), and an online interface. Technical superiority, distribution flexibility, scalability, transparency, usability and accessibility, integration and customization, and automation readiness are among the benefits over current tools. Strategic benefits for management include improved decision-making, efficiency gains, cross-disciplinary use, and compliance and audit readiness. The calculator ensures measurement results have well-quantified uncertainty, facilitates risk-based thinking in line with ISO/IEC 17025 requirements, reduces time spent on manual uncertainty spreadsheets, and reduces calculation errors by automating repetitive processes. It is applicable in multiple divisions and reduces training overhead. Table 1 presents some comparative advantages of this new program over conventional ones.

4. PROGRAM STRUCTURE

4.1. Architecture of the System

Streamlit is web-based and compatible with all browsers provides model equation entry and dynamic input fields. Analytical and Monte Carlo results panels are distinct. Automatic creation of PDF reports. The Streamlit framework is used to implement the application in Python, allowing for browser-based interaction without the need for local installation. In addition to choosing from preloaded measurement models or defining custom functions, users can enter measured values, correlation coefficients, and standard uncertainties. Figure 1 presents the program structure.

4.2. The Engine of Uncertainty Propagation

There are two modes of calculation: (a) Monte Carlo evaluation with user-specified iteration counts, producing histograms and statistical summaries; and (b) first-order GUM propagation, which computes combined standard uncertainty using sensitivity coefficients. Analytical calculations are validated in real time by Monte Carlo results.

4.3. Complex Modelling

Nonlinear models, for instance Planck-law fitting for radiation thermometry, are incorporated into the tool. Users can upload instrument spectral response data or specify an effective wavelength (λ_{eff}). Next, nonlinear propagation is carried out directly within the physical model by the calculator.

4.4. Implementation details

The engine uses numerical central differences with adaptive step size to compute sensitivity coefficients. Monte Carlo sampling draws independent normal deviates by default; correlation can be added via a covariance matrix and a Cholesky factorization. For instance, in radiation thermometry, numerical quadrature (trapezoidal rule on a user-supplied wavelength grid) computes the expected signal; inversion uses a robust scalar root finder over temperature.

4.5. Evaluation Cycle (simplified)

1) Read inputs (x, u, distributions, correlations).

2) Compute nominal outputs y = f(x).

3) Numerical sensitivities: $\boldsymbol{c_i = \frac{\partial y}{\partial x_i}}$

4) First-order $\boldsymbol{u_c^2 = \sum (c_i\, u_i)^2}$ (+ covariance terms).

5) Monte Carlo: draw N samples of x; evaluate y; compute mean, STDEV, percentiles.

6) Export GUI PNG, PDF report, and CSV (samples).

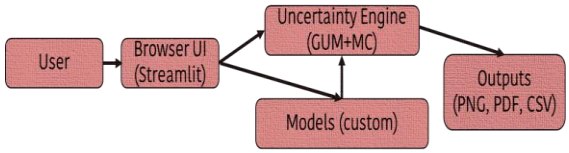

Figure 1. System architecture: browser UI interacts with the uncertainty engine and model modules. Outputs include PNG, PDF and CSV.

Table 1. Comparison with respect to conventional tools

ELEMENT	THIS SOFTWARE	CONVENTIONAL TOOLS
UNCERTAINTY	GUM + MC SIDE-BY-SIDE	ONE TECHNIQUE (USUALLY GUM ONLY)
MODELLING	LINEAR + NON-LINEAR POSSIBILITY, CUSTOM MODELS	PRIMARILY LINEAR
TRANSPARENCY	FORMULAS, SENSITIVITIES, VARIANCE BUDGET	BLACK-BOX OUTPUTS
EXTENSIBILITY	OPEN PYTHON CORE, CSVJSON I/O	RIGID FEATURES, CLOSED SOURCE
REPORTING	PNG GUI, PDF, CSV SAMPLES	MINIMAL TABLE, NO AUTOMATION
VALIDATION	MC VS. ANALYTICAL	RARELY IMPLEMENTED

4. CASE STUDY

Interferometric thickness measurements can achieve sub-micrometer accuracy, but the total uncertainty depends on laser wavelength calibration, fringe counting, air refractive index, mechanical alignment, and environmental stability. This Web-based tools can improve transparency and repeatability by coupling physics-based models with audit-ready reporting.

In metrology, interferometry is a technique that measures the thickness of thin films by examining how light waves interfere with reflections from a transparent film's top and bottom surfaces. A known refractive index is necessary for this procedure, which can be carried out with monochromatic or white-light sources. It enables accurate determination of the optical path difference and the thickness of the film. Interference, signal separation, and thickness calculation are all part of the process. Using the known refractive index and light wavelength, the thickness of the film is determined by calculating the optical path difference (OPD) between the interfering beams from the interference pattern.

A He–Ne source (λ = 632.8 nm) illuminates a Twyman–Green interferometer that measures a glass coupon's thickness inside a controlled chamber (T = 20.0 °C, P=1013.2 hPa, RH=45%). The effective air refractive index n=1.0002713 with standard uncertainty 2.0^{-07}. The model equation and nominal fringe count are shown below. Alignment residuals are modeled with σ =0.05 μm. Uncertainty is propagated via linearization around nominal values and validated by Monte Carlo sampling.

Agreement between analytical and Monte Carlo estimates confirms near-linearity of the model at the stated uncertainties. The budget indicates fringe counting and alignment dominate; wavelength and air-index contributions are smaller under the stated priors. The GUI documents assumptions and enables direct export of figures and a structured PDF.

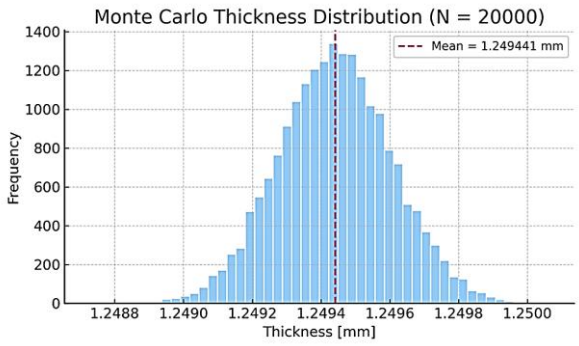

Figure 2. Monte Carlo Distribution yielded from thickness example execution.

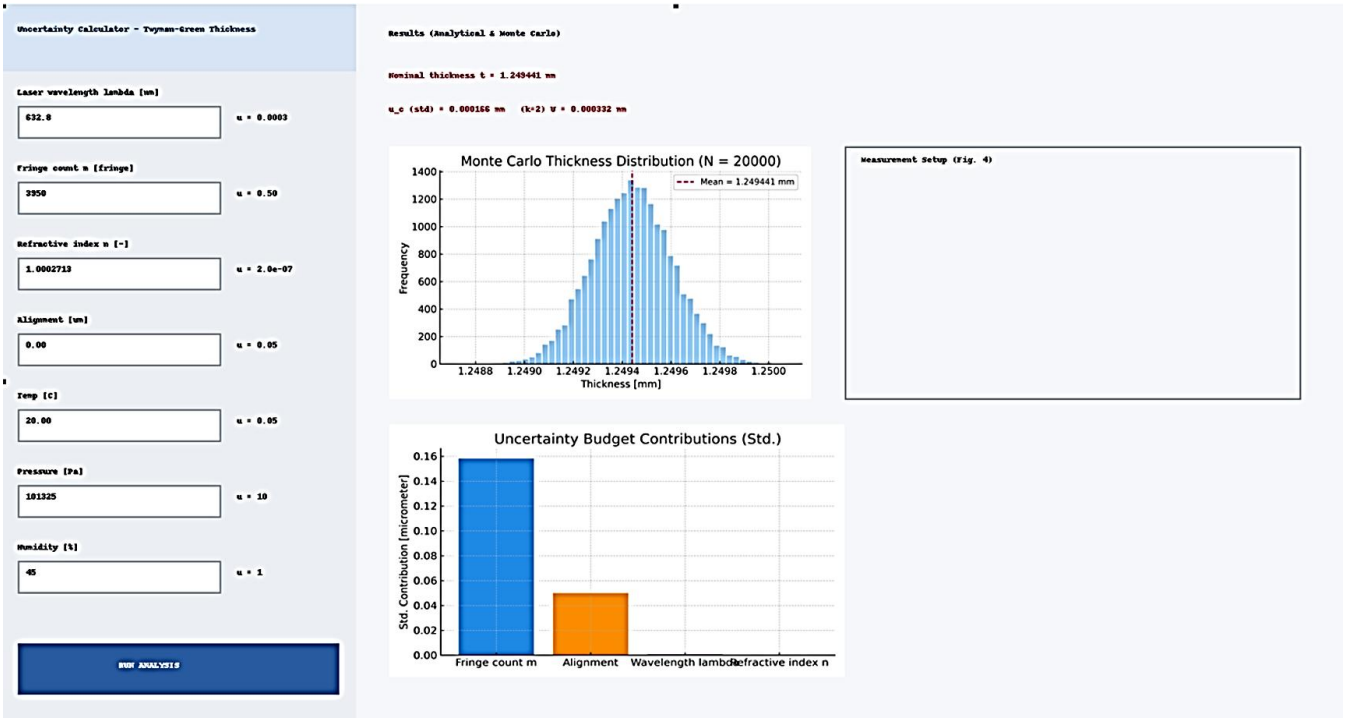

Figure 3. GUI of the program with thickness example implemented.

4.1. Measurement model, conditions and uncertainty calculator outcomes

$$t = \frac{\lambda}{2n}\left(m + \frac{\varphi}{2\pi}\right) \qquad (1)$$

where t is the thickness in nm, λ the wavelength (nm), n the refractive index, m the fringe order (integer), and φ the measured phase difference in radians.

- Inputs (example):
  - λ = 632.8 nm , σ_λ = 0.1 nm
  - n_{air}= 1.0002713
  - n = 1.500, σ_n = 0.005
  - m = 100 (exact integer)
  - φ = 1.234 rad, σ_φ = 0.02 rad
  - Monte Carlo samples : n = 20000
- Results:
  - Nominal thickness : t_0 = 21134.76004933929 nm
  - First order (numerical) $\sigma_t \approx$ 70.5315 nm (1σ)
  - Monte Carlo: *mean* ≈ 21135.344215809193 nm, *std* ≈ 70.3915 nm (1σ)
- Numerical partial derivatives (central differences):
  - $\partial t/\partial\lambda \approx$ 33.3988 nm/nm
  - $\partial t/\partial n \approx$ -14089.84 nm per unit n
  - $\partial t/\partial\varphi \approx$ 33.57108 nm per rad.

Because of the problem's modest nonlinearity (division by n and addition of φ/2π inside), Monte Carlo and first-order methods produce solutions that are similar but somewhat different; MC accounts for any slight sampling asymmetry.

The histogram aids in identifying any non-Gaussian characteristics or skewness in the t-distribution (in this case, it appears to be near-Gaussian). Figure 2 shows Monte Carlo chart of the presented example and figure 3 presents the GUI of the developed prototype (demo) uncertainty calculator, while this real-world example is implemented to demonstrate the applicability of the program.

5. CONCLUSIONS

The system provides quicker, error-free uncertainty evaluations by combining Monte Carlo simulation and the ISO GUM analytical method into a browser-based platform. It facilitates multidisciplinary testing, such as mechanical and optical metrology. Under ISO/IEC 17025, compliance-ready reports in PDF format can be tracked for audits. No licensing fees and flexibility to adjust to changing measurement requirements are made possible by the open-source foundation. Using this tool gives the metrology leader data-driven confidence in each measurement, strengthening their position.

The general Python-based graphical uncertainty calculator is a clever upgrade over conventional, fragmented tools. It offers a transparent, scientifically sound, and management-friendly solution that enables both analytical and simulation-based uncertainty evaluation on a single, user-friendly platform. Its open-source nature, which ensures long-term flexibility without licensing fees, makes it the ideal choice for modern metrology labs seeking efficiency, compliance, and traceability.

Combining Monte Carlo and GUM analysis, being web-based and installation-free, being multidisciplinary (mechanical, electrical, thermal, and optical), adhering to ISO/IEC 17025, eliminating spreadsheet errors, and being open-source with no license fees are some of its salient features.

Among the advantages are quicker, error-free evaluations, handling intricate, non-linear models, and creating audit-ready PDF reports; Fair assistance for R&D and manufacturing; flexible to the organization's requirements; boosts measurement accuracy.

Finally, this work presents a Python-based graphical uncertainty calculator that provides a flexible and scientifically sound way to measure uncertainty quantification, making it a useful tool for labs and research and development facilities.

Normal distributions and uncorrelated inputs are now assumed by the defaults. Future releases will reveal correlation matrices and other distributions (rectangular, triangular, and U-shaped) with on-the-fly inversion. Batch mode and an API endpoint will support automated certificate generation.

ACKNOWLEDGMENTS

Authors thank the “Research and Studies Center (RSC)” of SASO for their coordination and consideration of this work.

REFERENCES

[1] Bich, W., Cox, M. G., & Harris, P. M. (2006). Evolution of the "Guide to the Expression of Uncertainty in Measurement." *Metrologia*, *43*(4), S161–S166. https://doi.org/10.1088/0026-1394/43/4/S02

[2] Cox, M. G., & Siebert, B. R. L. (2006). The use of a Monte Carlo method for evaluating uncertainty and expanded uncertainty. *Metrologia*, *43*(4), S178–S188. https://doi.org/10.1088/0026-1394/43/4/S03

[3] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[4] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

[5] Joint Committee for Guides in Metrology. (2008). *Evaluation of measurement data — Guide to the expression of uncertainty in measurement* (JCGM 100:2008). https://www.bipm.org/documents/20126/2071204/JCGM_100_2008_E.pdf

[6] Joint Committee for Guides in Metrology. (2008). *Evaluation of measurement data — Supplement 1 to the "Guide to the expression of uncertainty in measurement" — Propagation of distributions using a Monte Carlo method* (JCGM 101:2008). https://www.bipm.org/documents/20126/2071204/JCGM_101_2008_E.pdf

[7] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter Notebooks—a publishing format for reproducible computational workflows. *ELPUB*, *2016*, 87–90. https://doi.org/10.4000/proceedings.613

[8] Lebigot, E. O. (2010). uncertainties: a Python package for calculations with uncertainties. *Journal of Open Research Software*, *8*(1), 14. https://doi.org/10.5334/jors.238

[9] McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 51–56). https://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf

[10] Millman, K. J., & Aivazis, M. (2011). Python for scientists and engineers. *Computing in Science & Engineering*, *13*(2), 9–12. https://doi.org/10.1109/MCSE.2011.36

[11] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html

[12] Ramirez, A., & Barnes, J. (2020). GUI development for scientific applications: A comparative study of PyQt and Tkinter. *Journal of Computational Science Education*, *11*(2), 45–58. https://doi.org/10.22369/jcse.2020.1102.03

[13] Siebert, B. R. L., & Sommer, K. D. (2004). New developments of the GUM and Monte Carlo techniques. *Technisches Messen*, *71*(2), 67–80. https://doi.org/10.1524/teme.71.2.67.27066

[14] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... van Mulbregt, P. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[15] Willink, R. (2006). On using the Monte Carlo method to calculate uncertainty intervals. *Metrologia*, *43*(6), L39–L42. https://doi.org/10.1088/0026-1394/43/6/N01