SoftwareX 11 (2020) 100450

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Pygpc: A sensitivity and uncertainty analysis toolbox for Python

Konstantin Weise [a,b,*], Lucas Poßner [c], Erik Müller [c], Richard Gast [a], Thomas R. Knösche [a,d]

[a] Methods and Development Group Brain Networks, Max Planck Institute for Human Cognitive and Brain Sciences, Stephanstr. 1a, 04103 Leipzig, Germany
[b] Technische Universität Ilmenau, Advanced Electromagnetics Group, Helmholtzplatz 2, 98693 Ilmenau, Germany
[c] Leipzig University of Applied Sciences, Institute for Electronics and Biomedical Information Technology, Wächterstr. 13, 04107 Leipzig, Germany
[d] Technische Universität Ilmenau, Institute of Biomedical Engineering and Informatics, Gustav-Kirchhoff-Straße 2, 98693 Ilmenau, Germany

ARTICLE INFO

Article history:
Received 31 January 2020
Received in revised form 6 March 2020
Accepted 6 March 2020

Keywords:
Sensitivity analysis
Uncertainty analysis
Polynomial chaos

ABSTRACT

We present a novel Python package for the uncertainty and sensitivity analysis of computational models. The mathematical background is based on the non-intrusive generalized polynomial chaos method allowing one to treat the investigated models as black box systems, without interfering with their legacy code. Pygpc is optimized to analyze models with complex and possibly discontinuous transfer functions that are computationally costly to evaluate. The toolbox determines the uncertainty of multiple quantities of interest in parallel, given the uncertainties of the system parameters and inputs. It also yields gradient-based sensitivity measures and Sobol indices to reveal the relative importance of model parameters.

Code metadata

Current code version	*v0.27*
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_7
Code Ocean compute capsule	–
Legal Code License	*BSD 3-clause/GPL-3.0*
Code versioning system used	*GitHub*
Software code languages, tools, and services used	*Python, C++, CUDA, OpenMP, Travis CI*
Compilation requirements, operating environments & dependencies	*OS: Linux, Windows, MacOS*
If available Link to developer documentation/manual	https://pygpc.readthedocs.io/en/latest/
Support email for questions	kweise@cbs.mpg.de

1. Motivation and significance

Computational modeling is crucial for the investigation of a broad range of natural and artificial systems, such as technical appliances, climate and weather systems, economical and social systems, as well as the brain. The output of such models depends on parameters that are usually not exactly known for various reasons, but which can be described by probability density functions. Therefore, it is crucial to know the degree of uncertainty (quantified by its probability density function) of the output variables, given the uncertainties of the parameters. Furthermore, we sought to reveal how sensitive the output is towards each parameter, as this tells us which parameters are the most important to be determined accurately and which are less critical.

In order to answer these questions, the parameter space is usually sampled in a random or systematic fashion, possibly guided by known or assumed probability density functions. Classical methods, such as naïve Monte Carlo approaches or Brute-force search [1], would require producing many samples of the computationally expensive models. However, in many relevant models the parameter spaces are large, the model evaluations can be time consuming, and these methods quickly generate prohibitive computational costs. Alternative approaches that seek to reduce the computational effort, such as worst case evaluation [2] or linearization [3], are much more limited in their conclusions.

* Corresponding author at: Methods and Development Group Brain Networks, Max Planck Institute for Human Cognitive and Brain Sciences, Stephanstr. 1a, 04103 Leipzig, Germany.
E-mail address: kweise@cbs.mpg.de (K. Weise).

https://doi.org/10.1016/j.softx.2020.100450

For these reasons, pygpc is based on the more efficient, non-intrusive, generalized polynomial chaos method (gPC) [4]. It is capable of analyzing black-box systems by virtue of a highly efficient meta-model of the original transfer function, from which the stochastic properties and sensitivities of the quantities of interest (QOI) are derived. gPC in general has been applied in a variety of applications such as computational fluid dynamics [5–7], heat transfer [8,9], multibody dynamics [10,11], and robust design optimization [12].

Pygpc is a Python based gPC library, with high performant C and CUDA-C extensions, offering a user-friendly interface to analyze one's own models written in either Python, Matlab, or any shell interfaceable program. Its efficient parallel CPU/GPU implementation allows the analysis of very complex models, including those containing discontinuities, on computer hardware of different performance ranging from laptops to large compute clusters. The implemented sensitivity analysis allows the identification of the most important parameters of the model under investigation and considerably accelerates prototyping and model analysis. Existing uncertainty quantification software packages, such as the UQ Toolkit (UQTK) [13,14] (https://github.com/sandialabs/UQTk), the MIT Uncertainty Quantification (MUQ) C++ Library (http://muq.mit.edu/), and UQLab [15] (https://www.uqlab.com/), do not offer this combination of advanced features.

Thus far, we have applied pygpc in the frameworks of non-destructive testing [16] and neuroscience [17–20]. However, it can be applied to a broad range of scientific and engineering fields.

2. Software description

The core concept of the gPC method is to find a functional dependence between the random variables $\boldsymbol{\xi}$ (input parameters) and the quantity of interest $\boldsymbol{q}$ by means of an orthogonal polynomial basis Ψ:

$$\boldsymbol{q}(\boldsymbol{\xi}) \approx \sum_{\boldsymbol{\alpha}\in\mathcal{A}} \boldsymbol{u}_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) . \tag{1}$$

The functions $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) = \prod_{i=1}^{d} \psi_{\alpha_i}^{i}(\xi_i)$ are the joint polynomial basis functions of the gPC. They are composed of polynomials $\psi_{\alpha_i}^{i}(\xi_i)$ that are separately defined for each random variable. The polynomials are chosen to be orthogonal in the normed spaced induced by the probability density functions (PDFs) $p_i(\sigma_i)$ [21]. The multi-index $\boldsymbol{\alpha}$ of the joint basis function includes the degrees of the individual polynomials and the set $\mathcal{A}$ of cardinality N_c contains the multi-indices of the chosen basis.

Pygpc offers the possibility to use a regression or a quadrature approach to obtain the gPC coefficients $\boldsymbol{u}_{\boldsymbol{\alpha}}$. For example, writing (1) in matrix form yields the system of equations solved in the regression approach:

$$[\Psi][\mathbf{U}] = [\mathbf{Q}] , \tag{2}$$

where $[\Psi]$ is the gPC matrix of size $[N_s \times N_c]$, where N_s denotes the number of samples (model evaluations) and N_c denotes the number of basis functions and gPC coefficients, respectively. $[\mathbf{U}]$ is the coefficient matrix of size $\left[N_c \times N_q\right]$ containing the gPC coefficients of the N_q QOIs, and $[\mathbf{Q}]$ is the solution matrix of size $\left[N_s \times N_q\right]$ containing the results of the model evaluations. Solving (2) for $[\mathbf{U}]$ yields the polynomial surrogates of the QOIs as a function of the random input parameters $\boldsymbol{\xi}$. This enables computationally efficient investigations of their statistics and sensitivities. For example, the expectation (i.e., mean) $\boldsymbol{\mu}$ and variance $\boldsymbol{\nu}$ are determined by:

$$\boldsymbol{\mu} = \boldsymbol{u}_{\boldsymbol{\alpha}_0} \tag{3}$$

$$\boldsymbol{\nu} = \sum_{\boldsymbol{\alpha}\in\mathrm{A}\setminus\boldsymbol{\alpha}_0} (\boldsymbol{u}_{\boldsymbol{\alpha}})^2 . \tag{4}$$

The Sobol indices $\boldsymbol{S}_i^{(\upsilon)}$ decompose the total variance $\boldsymbol{\nu}$ of the quantity of interest into components that can be attributed to individual random variables $\xi_{\boldsymbol{i}}$ or combinations thereof [22,23]. For each $\boldsymbol{S}_i^{(\upsilon)}$, only the squared coefficients $\boldsymbol{u}_{\boldsymbol{\alpha}}^2$, whose multi-indices $\boldsymbol{\alpha}$ belong to the set A_i with non-zero values only for the ξ_i of interest are considered.

$$\boldsymbol{S}_i^{(\upsilon)} = \frac{1}{\nu} \sum_{\boldsymbol{\alpha}\in A_i} (\boldsymbol{u}_{\boldsymbol{\alpha}})^2 \tag{5}$$

The global derivative-based sensitivity coefficients $\boldsymbol{S}_i^{(\partial)}$ are measures of the average change of the quantity of interest with respect to the ith random variable. They are determined by means of the gPC-coefficients and the corresponding partial derivatives of the basis functions [24]:

$$\boldsymbol{S}_i^{(\partial)} = \mathbb{E}\left[\frac{\partial \boldsymbol{q}(\boldsymbol{\xi})}{\partial \xi_i}\right] = \sum_{\boldsymbol{\alpha}\in\mathcal{A}_i} \boldsymbol{u}_{\boldsymbol{\alpha}} \int_{\Theta} \frac{\partial \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})}{\partial \xi_i} p(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi} \tag{6}$$

Finally, the PDFs of the QOIs are obtained by sampling the gPC surrogate using traditional MC methods. This is possible without much computational effort because the evaluation of the polynomial description is far less time consuming than the original computation.

The foundation of gPC is the construction of a basis capable of emulating model behavior while choosing the most informative sampling points in order to determine the associated gPC coefficients. There are several possibilities published in the literature, and developed by the authors of pygpc, to increase the efficiency of the gPC method, which will be discussed in more detail below.

2.1. Software architecture

A simplified overview of the software architecture of pygpc is given in Fig. 1. After the model is set up by the user, using the *Model* class, the uncertainty problem is defined by initializing the *Problem* class. This is done by assigning the random parameters using the *RandomParameter* class. The gPC algorithm is the core element and determines how the surrogate model is constructed. It handles the construction of the basis and manages the sampling in the *Grid* class. Subsequently, the *Computation* class calls the model and runs the calculations necessary to determine the gPC coefficients in the *GPC* class. In case of discontinuous transfer functions, the random space is divided into sections and the classifier assigns the sampling points to the different sub-regions. The analysis is completed by reaching a certain approximation order or by satisfying a previously defined convergence criterion. Thereafter, all metadata is saved in a gPC *Session* object, whereas the gPC coefficients, simulation results, etc., are saved in an associated file using the hierarchical data format (HDF5, www.hdfgroup.org).

2.2. Software functionalities

Pygpc offers the possibility to assign uniform, beta, gamma, and normal distributions to the random variables. The surrogate model is constructed using the corresponding families of orthogonal polynomials, i.e., Legendre, Jacobi, Laguerre, and Hermite polynomials. Besides models written in Python, it is possible to investigate MATLAB models by installing the MATLAB Engine API for Python [25].

An important feature of pygpc is the large variety of implemented gPC algorithms. In general, there is the possibility to

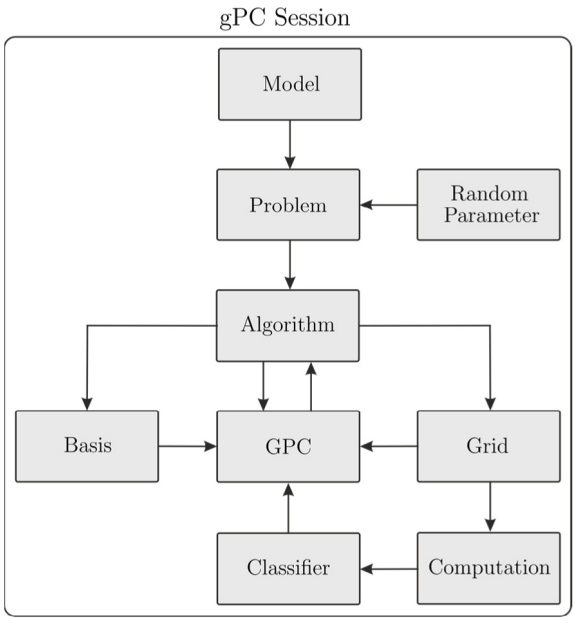

Fig. 1. Simplified software architecture of pygpc, showing the basic classes and their interactions.

either use *static* algorithms [17] with a predefined approximation order and number of sampling points, or *adaptive* algorithms [19], which successively construct the basis and the grid of sampling points until a certain error criterion is met. So far, standard random sampling and latin hypercube sampling (LHS) [26,27] are implemented, which will be extended by further sampling methods in the future.

In pygpc, it is possible to use the gradient of the transfer function in order to enhance the accuracy and efficiency of gPC [28,29]. Besides that, different solvers can be chosen to determine the gPC coefficients including, for example, classical smooth $\ell 2$ minimizers, like the Moore–Penrose pseudo inverse, or sparse $\ell 1$ minimization, like least-angle regression (LARS), or orthogonal matching pursuit (OMP) [30]. The latter, in combination with our random sampling methods, constitutes a compressed sensing scheme [30,31].

As the number of random variables increases, the curse of dimensionality can make analysis considerably more difficult or even impossible. To prevent this, pygpc allows the adaptive reduction of the number of random variables by re-parameterizing the original model. An optimal rotation and reduction of the gPC basis [32,33] is performed by identifying the principle components of the Jacobian of the QOI.

When the quantity of interest depends discontinuously on the input parameters, the standard gPC may lack sufficient accuracy because it makes use of polynomials defined in the entire domain. To overcome this, pygpc supports a multi-element gPC approach (MEGPC), which includes a domain decomposition method based on k-means clustering [34]. The latter subdivides the random space and provides input data for the Multi-Layer Perceptron classifier [35–37], in order to assign new sampling points to the corresponding domain. Both, the clusterer and the classifier can be replaced by different algorithms, which are implemented in the machine learning package scikit-learn [38].

After the gPC is finished and the surrogate is constructed, the model is post-processed by determining the output PDFs and the first four central moments, i.e. mean, standard deviation, skewness, and kurtosis of the QOI(s). In a subsequent sensitivity analysis, the Sobol indices [22,23] and the global derivative based sensitivity coefficients [9] are determined for every QOI.

The construction of the gPC matrix, its solution to determine the coefficient matrix, and the multiplication of the gPC matrix with the coefficient matrix are the processes with the highest computational cost. This is especially true when the surrogate consists of a high number of basis functions or a high number of surrogate model evaluations have to be performed. To increase the performance of pygpc, we developed a highly efficient C++ extension. We used OpenMP and CUDA to perform the computationally complex creation of the (possibly large) gPC matrix in parallel. Depending on the hardware used, the performance could be significantly increased compared to pure Python implementations.

2.3. Sample code snippet analysis

As a first step of every gPC analysis, the user is required to provide a wrapper of their model using the *AbstractModel* class (Fig. 2). In a second step, the gPC session is set up by loading the model, defining the uncertain parameters and their distributions (problem definition), choosing an algorithm, and running the analysis by starting the gPC session. The frontend of pygpc has a modular structure of the form: model → problem → algorithm → session. In this way, the model can be easily replaced by another one or the problem can be redefined by assigning different properties to the random variables or by defining some of them as constants. Alternatively, the algorithm to construct the surrogate model can be replaced with little effort.

3. Illustrative example

In biological systems, large inter-individual differences translate to considerable uncertainties in the model parameters. A typical problem in neuroscience is to link unknown structural parameters of the brain to recordings of brain activity [39–43]. The Jansen–Rit neural mass model (NMM) is a non-linear model for the dynamic interaction between two excitatory and one inhibitory population of neurons within a cortical column [44]. It is described by a set of six nonlinearly coupled differential equations and has been previously used to infer the connection strength and direction between brain areas from electrophysiological recordings of macroscopic brain activity [45,46]. To link this model to experimental data, an understanding of the model's parameter dependencies is essential. In this example, we examine the input/output behavior of a Jansen–Rit NMM (Fig. 3b). We varied the amplitude and the frequency of the external stimulation as beta distributed random variables (Fig. 3a). The QOI is defined as the dominant frequency $\hat{f}_{PC}$ of the post-synaptic potential of the pyramidal cells, i.e. the frequency with the highest power-spectral-density. The model was set up using the Python package PyRates [47]. Since the model behavior shows rapid phase transitions (Fig. 3d), we used a MEGPC approach to approximate its behavior. The domain was split into three domains covering the low (blue), medium (green) and high frequency regions (red). The mean and the standard deviation of the dominant frequency in the parameter space under investigation, considering the probability densities of the input parameters, were 10.806 Hz and 0.305 Hz, respectively. The smooth transition between the low and high frequency domain ($f \approx 11$ Hz) is approximated by the gPC as a step. Overall, the gPC approximated the complex model behavior quite well. The overall difference between the original model and the gPC was about 5% (Fig. 3c). It was computed from an independent evaluation dataset of size N using the normalized root mean square deviation (NRMSD) between the original model solutions q_i and the gPC approximations $\tilde{q}_i$:

$$NRMSD = \frac{1}{N} \frac{\sqrt{\sum_{i=1}^{N} \left(\tilde{q}_i - q_i\right)^2}}{\max(\boldsymbol{q}) - \min(\boldsymbol{q})} \tag{7}$$

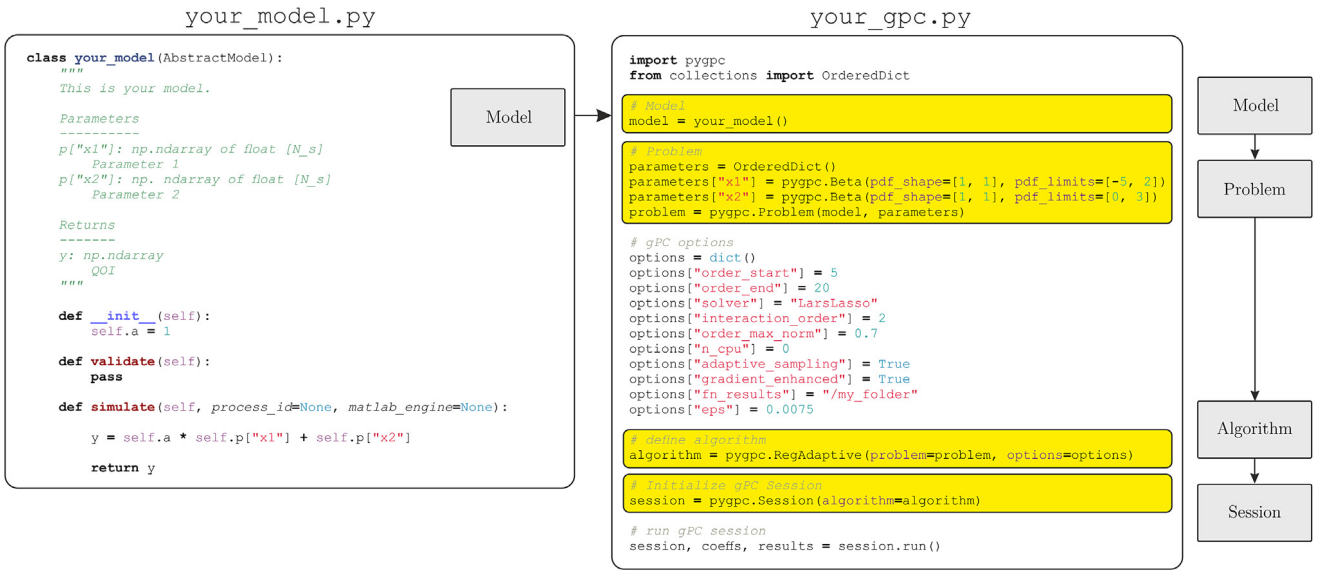

Fig. 2. Example analysis and overall structure of the frontend of pygpc; (left): setting up the model; (right): setting up the associated gPC session.

Table 1
Sobol indices and global derivative based sensitivity indices of the Jansen–Rit NMM.

Parameters	Sobol indices	Global derivatives
α	9.1e−4	−0.0019 V/V
f	0.4805	0.4246 V/Hz
$\alpha \cdot f$	0.5192	–

This can be explained by the higher probability density of the input parameters in the center of the parameter space, which further emphasizes this critical region. The results of the sensitivity analysis are shown in Table 1. Both sensitivity measures demonstrate that the dominant frequency is insensitive to the stimulation amplitude α, but very sensitive to the stimulation frequency f. The high Sobol coefficient of second order in combination with the coefficients of first order indicates a pronounced discontinuity in the parameter space.

4. Impact

Data errors resulting from, for example, errors in input variables or model parameters belong to the largest sources of error besides model and numerical errors. In most situations, the model parameters cannot be exactly specified, due to limitations in the available experimental data or because of inherent case-to-case variability of the systems studied. Inaccurate knowledge of the model parameters may lead to considerable differences between the studied real world system and the numerical simulations. The development process of new models should therefore always be accompanied by a thorough sensitivity analysis to investigate and quantify its stability and robustness. Those analyses have to run alongside the modeling process and should not become the main task of the modeler. The black-box character of pygpc allows rapid integration of user defined models without the need to make time-consuming adjustments. The combination of applying projection techniques together with state of the art $\ell 1$ minimization allows the computation of sparse gPC representations that counteract the curse of dimensionality, thus offering the possibility of investigating high dimensional systems. Moreover, the implemented MEGPC allows the analysis of discontinuous transfer functions, which are very common in real world systems.

In engineering, the identification of the most important sources of uncertainty, prior to production, makes it possible to decrease the number of iterations during prototyping. This shortens development time and increases cost efficiency.

Besides that, pygpc can be used to validate simulations against real world measurements. A comprehensive validation study must carefully take into consideration both experimental and computational uncertainty ranges. The latter are inherently affected by measurement errors and system imperfections, which are typically represented using error bars. The post-processing routines of pygpc allow the computation of the same measures for the numerical model, making measurements comparable to simulations. A practical example, where pygpc was applied, was presented for non-destructive Lorentz force eddy current testing [16,48]. By means of pygpc, it was possible to quantify the impact of multiple unknown input parameters to cost-efficiently improve the laboratory setup in terms of reliability and reproducibility and to validate the simulation results [16].

Another area of application is reliability and risk analysis, where the aim is to determine the probabilities and associated parameter combinations of the system where certain critical values or operating thresholds are exceeded.

Besides the aforementioned applications, the derived surrogates can be used to optimize the model behavior. In this case, the QOI is used to compute the goal function to be minimized. After applying gPC, the optimization may be performed on the computational efficient polynomial surrogate without the need to run expensive model calculations.

5. Conclusions

The presented software can be used in numerous areas of application. We demonstrated the capabilities of pygpc on a neuronal mass model with discontinuities. It was possible to construct a surrogate model of the complex transfer function with an accuracy of about 95%. With the help of sensitivity analysis it was possible to get insight into the parameter dependencies and to identify the most important parameters influencing the dominant frequency.

Our goal is to provide a versatile tool for efficient uncertainty and sensitivity analysis of black-box systems. Over the next years, it will be continuously maintained and further developed

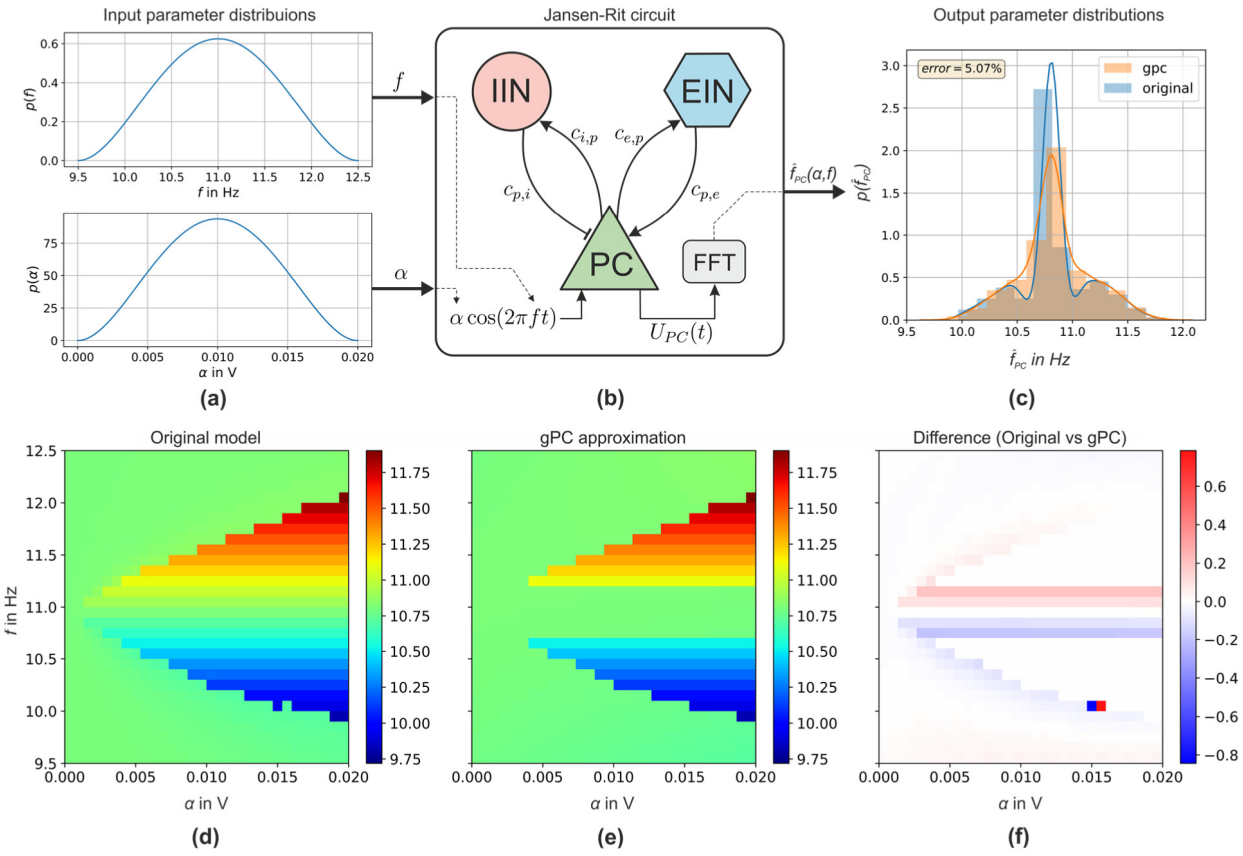

Fig. 3. Illustrative example of pygpc analyzing a Jansen–Rit type neuronal mass model in terms of its input/output behavior; (a) beta distributed input parameters with shape parameters $p = q = 3$; α is the amplitude and f is the frequency of the external stimulation; (b) Jansen–Rit neuronal mass model incorporating inhibitory interneurons (IIN), excitatory interneurons (EIN) and pyramidal cells (PC); (c) probability density function of the dominant frequency $\hat{f}_{PC}$ (highest power density) of the postsynaptic potential of the pyramidal cells; (d) Dominant frequency as a function of α and f obtained from the original model; (e) gPC approximation of the dominant frequency; (f) absolute difference between original model and gPC approximation . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

by incorporating new sampling methods, matrix solvers, post-processing routines and adaptive algorithms to further enhance its efficacy. The modular structure of pygpc allows for contribution and fast implementation in any of the aforementioned fields.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by the German Science Foundation (DFG) (grant number WE 59851/2) and the NVIDIA Corporation, Germany (donation of two Titan Xp graphics cards to KW and TK). We acknowledge support for the Article Processing Charge by the German Research Foundation (DFG) and the Open Access Publication Fund of the Technische Universität Ilmenau.

References

[1] Salvador R, Ramirez F, V'yacheslavovna M, Miranda PC. Effects of tissue dielectric properties on the electric field induced in tDCS: A sensitivity analysis. In: 2012 annual international conference of the IEEE engineering in medicine and biology society. p. 787–90. http://dx.doi.org/10.1109/EMBC.2012.6346049.

[2] Thielscher A, Opitz A, Windhoff M. Impact of the gyral geometry on the electric field induced by transcranial magnetic stimulation. Neuroimage 2011;54(1):234–43. http://dx.doi.org/10.1016/j.neuroimage.2010.07.061.

[3] Santos L, Martinho M, Salvador R, Wenger C, Fernandes SR, Ripolles O, et al. Evaluation of the electric field in the brain during transcranial direct current stimulation: A sensitivity analysis. In: 2016 38th annual international conference of the IEEE engineering in medicine and biology society (EMBC). Orlando, FL, 2016. p. 1778–81. http://dx.doi.org/10.1109/EMBC.2016.7591062.

[4] Kaintura A, Dhaene T, Spina D. Review of polynomial chaos-based methods for uncertainty quantification in modern integrated circuits. Electronics 2018;7(3):30. http://dx.doi.org/10.3390/electronics7030030.

[5] Knio OM, Le Maitre OP. Uncertainty propagation in CFD using polynomial chaos decomposition. Fluid Dyn Res 2006;38(9):616–40. http://dx.doi.org/10.1016/j.fluiddyn.2005.12.003.

[6] Xiu D, Karniadakis GE. Modeling uncertainty in flow simulations via generalized polynomial chaos. J Comput Phys 2003;187(1):137–67. http://dx.doi.org/10.1016/S0021-9991(0300092-5).

[7] Hosder S, Perez R, Walters R. A non-intrusive polynomial chaos method for uncertainty propagation in CFD simulations. In: Proceedings of the 44th AiAA aerospace sciences meeting and exhibit, Vol. 891. 2006, p. 1–19. http://dx.doi.org/10.2514/6.2006-891.

[8] Wan X, Xiu D, Karniadakis GE. Modeling uncertainty in three-dimensional heat transfer problems. In: Sundén B, Brebbia CA, Mendes A, editors. Advanced computational methods in heat transfer VIII. WIT transactions on engineering sciences, vol. 46, 2004, p. 1–11. http://dx.doi.org/10.2495/HT040021.

[9] Xiu D, Karniadakis GE. A new stochastic approach to transient heat conduction modeling with uncertainty. Int J Heat Mass Transfer 2003;46(24):4681–93. http://dx.doi.org/10.1016/S0017-9310(0300299-0).

[10] Sandu A, Sandu C, Ahmadian M. Modeling multibody systems with uncertainties. Part I: Theoretical and computational aspects. Multibody Syst Dyn 2006;15(4):369–91. http://dx.doi.org/10.1007/s11044-006-9007-5.

[11] Sandu C, Sandu A, Ahmadian M. Modeling multibody systems with uncertainties. Part II: Numerical applications. Multibody Syst Dyn 2006;15(3):241–62. http://dx.doi.org/10.1007/s11044-006-9008-4.
[12] Zein S. A polynomial chaos expansion trust region method for robust optimization. Commun Comput Phys 2013;14(2):412–24. http://dx.doi.org/10.4208/cicp.260512.260912a.
[13] Debusschere BJ, Najm HN, Pébay PP, Knio OM, Ghanem RG, Le Maitre OP. Numerical challenges in the use of polynomial chaos representations for stochastic processes. SIAM J Sci Comput 2004;26(2):698–719. http://dx.doi.org/10.1137/S1064827503427741.
[14] Debusschere B, Sargsyan K, Safta C, Chowdharyet K. The uncertainty quantification toolkit (UQTk). In: Ghanem RG, Higdon D, Owhadi H, editors. Handbook of uncertainty quantification. Cham: Springer International Publishing; 2016, p. 1807–27. http://dx.doi.org/10.1007/978-3-319-12385-1_56.
[15] Marelli S, Sudret B. UQLab: A framework for uncertainty quantification in matlab. In: Beer Michael, Au Siu-Kui, Hall Jim W. editors. Second international conference on vulnerability and risk analysis and management (ICVRAM) and the sixth international symposium on uncertainty, modeling, and analysis (ISUMA). Liverpool, UK, 2014. p. 2554-63. http://dx.doi.org/10.1061/9780784413609.257.
[16] Weise K, Carlstedt M, Ziolkowski M, Brauer H. Uncertainty analysis in lorentz force eddy current testing. IEEE Trans Magn 2016;52(3). 6200104. http://dx.doi.org/10.1109/TMAG.2015.2480046.
[17] Weise K, Di Rienzo L, Brauer H, Haueisen J, Toepfer H. Uncertainty analysis in transcranial magnetic stimulation using non-intrusive polynomial chaos expansion. IEEE Trans Magn 2015;51(7). 5000408. http://dx.doi.org/10.1109/TMAG.2015.2390593.
[18] Codecasa L, Di Rienzo L, Weise K, Gross S, Haueisen J. Fast MOR-based approach to uncertainty quantification in transcranial magnetic stimulation. IEEE Trans Magn 2016;52(3). 7200904. http://dx.doi.org/10.1109/TMAG.2015.2475120.
[19] Saturnino GB, Thielscher A, Madsen KH, Knösche TR, Weise K. A principled approach to conductivity uncertainty analysis in electric field calculations. NeuroImage 2019;188(1):821–34. http://dx.doi.org/10.1016/j.neuroimage.2018.12.053.
[20] Weise K, Numssen O, Thielscher A, Hartwigsen G, Knoesche T. A novel approach to localize cortical TMS effects. Neuroimage 2020;209. 116486. http://dx.doi.org/10.1016/j.neuroimage.2019.116486.
[21] Askey R, Wilson JW. Some basic hypergeometric orthogonal polynomials that generalize Jacobi-polynomials. Mem Amer Math Soc 1985;54(319):1–55, (ISBN: 978-1-4704-0732-2).
[22] Sobol IM. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. Math Comput Simulation 2001;55(1–3):271–80.
[23] Sudret B. Global sensitivity analysis using polynomial chaos expansions. Reliab Eng Syst Saf 2008;93(7):964–79. http://dx.doi.org/10.1016/j.ress.2007.04.002.
[24] Xiu D. Fast numerical methods for stochastic computations: A review. Commun Comput Phys 2009;5(2–4):242–72.
[25] MATLAB. Version 9.7 (R2019b). Natick, Massachusetts: The MathWorks Inc.; 2019.
[26] Mckay MD, Beckman RJ, Conover WJ. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 1979;21(2):239. http://dx.doi.org/10.2307/1268522.
[27] Loh WL. On Latin hypercube sampling. Ann Statist 1996;24(5):2058–80. http://dx.doi.org/10.1214/aos/1069362310.
[28] Jakeman JD, Eldred MS, Sargsyan K. Enhancing $\ell 1$ -minimization estimates of polynomial chaos expansions using basis selection. J Comput Phys 2015;289:18–34. http://dx.doi.org/10.1016/j.jcp.2015.02.025.
[29] Peng J, Hampton J, Doostan A. On polynomial chaos expansion via gradient-enhanced $\ell 1$-minimization. J Comput Phys 2016;310(1):440–58. http://dx.doi.org/10.1016/j.jcp.2015.12.049.
[30] Hampton J, Doostan A. Compressive sampling methods for sparse polynomial chaos expansions. In: Ghanem RG, Higdon D, Owhadi H, editors. Handbook of uncertainty quantification. Cham: Springer International Publishing; 2016, p. 827–55. http://dx.doi.org/10.1007/978-3-319-12385-1_67.
[31] Rauhut H. Compressive sensing and structured random matrices. In: Fornasier Massimo, editor. Theoretical foundations and numerical methods for sparse recovery. Radon series on computational and applied mathematics, vol. 9, Berlin: deGruyter; ISBN: 9783110226157, 2010, p. 1–92.
[32] Tipireddy R, Ghanem R. Basis adaptation in homogeneous chaos spaces. J Comput Phys 2014;259(1):304–17. http://dx.doi.org/10.1016/j.jcp.2013.12.009.
[33] Tsilifis P, Huan X, Safta C, Sargsyan K, Lacaze G, Oefelein JC, et al. Compressive sensing adaptation for polynomial chaos expansions. J Comput Phys 2019;380(1):29–47. http://dx.doi.org/10.1016/j.jcp.2018.12.010.
[34] Lloyd S. Least squares quantization in PCM. IEEE Trans Inform Theory 1982;28(2):129–37. http://dx.doi.org/10.1109/TIT.1982.1056489.
[35] Hinton GE. Connectionist learning procedures. Artificial Intelligence 1989;40(1–3):185–234. http://dx.doi.org/10.1016/0004-3702(8990049-0).
[36] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the 13th international conference on artificial intelligence and statistics. 2010. p. 249–56.
[37] He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the 2015 ieee international conference on computer vision (ICCV). 2015, p. 1026–34. http://dx.doi.org/10.1109/ICCV.2015.123.
[38] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in python. J Mach Learn Res 2011;12(1):2825–30, url: http://dl.acm.org/citation.cfm?id=1953048.2078195.
[39] Honey CJ, Sporns O, Cammoun L, Gigandet X, Thiran JP, Meuli R, et al. Predicting human resting-state functional connectivity from structural connectivity. Proc Natl Acad Sci USA 2009;106(6):2035–40. http://dx.doi.org/10.1073/pnas.0811168106.
[40] Bojak I, Oostendorp TF, Reid AT, Kötter R. Connecting mean field models of neural activity to EEG and fMRI data. Brain Topogr 2010;23(2):139–49. http://dx.doi.org/10.1007/s10548-010-0140-3.
[41] Friston KJ, Dolan RJ. Computational and dynamic models in neuroimaging. NeuroImage 2010;52(3):752–65. http://dx.doi.org/10.1016/j.neuroimage.2009.12.068.
[42] Breakspear M. Dynamic models of large-scale brain activity. Nature Neurosci 2017;20(3):340–52. http://dx.doi.org/10.1038/nn.4497.
[43] Deco G, Jirsa VK, Robinson PA, Breakspear M, Friston K. The dynamic brain: From spiking neurons to neural masses and cortical fields. PLoS Comput Biol 2008;4(8). e1000092. http://dx.doi.org/10.1371/journal.pcbi.1000092.
[44] Jansen BH, Rit VG. Electroencephalogram and visual evoked potential generation in a mathematical model of coupled cortical columns. Biol Cybernet 1995;73(4):357–66. http://dx.doi.org/10.1007/BF00199471.
[45] David O, Kiebel SJ, Harrison LM, Mattout J, Kilner JM, Friston KJ. Dynamic causal modeling of evoked responses in EEG and MEG. NeuroImage 2006;30(4):1255–72. http://dx.doi.org/10.1016/j.neuroimage.2005.10.045.
[46] Sotero RC, Trujillo-Barreto NJ. Biophysical model for integrating neuronal activity, EEG, fMRI and metabolism. NeuroImage 2008;39(1):290–309. http://dx.doi.org/10.1016/j.neuroimage.2007.08.001.
[47] Gast R, Rose D, Salomon C, Möller HE, Weiskopf N, Knösche TR. PyRates—A Python framework for rate-based neural simulations. PloS one 2019;14(12). e0225900. http://dx.doi.org/10.1371/journal.pone.0225900.
[48] Brauer H, Ziolkowski M, Uhlig RP, Zec M, Weise K, Carlstedt M. Motion-induced eddy current techniques for non-destructive testing and evaluation. Control, robotics and sensors ser, 1st ed.. Stevenage: Institution of Engineering & Technology; ISBN: 9781785612169, 2018.