SoftwareX 11 (2020) 100419

Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

TWINKLE: A digital-twin-building kernel for real-time computer-aided engineering

V. Zambrano, R. Rodríguez-Barrachina, S. Calvo, S. Izquierdo *

Instituto Tecnológico de Aragón - ITAINNOVA, C/ María de Luna 7-8, 50018 Zaragoza, Spain

ARTICLE INFO

Article history:
Received 7 March 2019
Received in revised form 23 October 2019
Accepted 4 February 2020

Keywords:
Model order reduction
PARAFAC
Machine learning
Data analysis
Tensor decomposition

ABSTRACT

TWINKLE is a library for building families of solvers to perform Canonical Polyadic Decomposition (CPD) of tensors. The common characteristic of these solvers is that the data structure supporting the tuneable solution strategy is based on a Galerkin projection of the phase space. This allows processing and recovering tensors described by highly sparse and unstructured data. For achieving high performance, TWINKLE is written in C++ and uses the Armadillo open source library for linear algebra and scientific computing, based on LAPACK (Linear Algebra PACKage) and BLAS (Basic Linear Algebra Subprograms) routines. The library has been implemented keeping in mind its future extensibility and adaptability to fulfil the different users' needs in academia and industry regarding Reduced Order Modelling (ROM) and data analysis by means of tensor decomposition. It is especially focused on post-processing data from Computer-Aided-Engineering (CAE) simulation tools.

Code metadata

Current code version	v0.2
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_64
Legal Code License	Dual: Commercial and GNU GPL 3
Code versioning system used	Tortoise SVN
Software code languages, tools, and services used	C++, Armadillo, BLAS, LAPACK
Compilation requirements, operating environments & dependencies	BLAS, LAPACK
If available Link to developer documentation/manual	
Support email for questions	sizquierdo@itainnova.es

1. Motivation and significance

Digital Twins are virtual representations of reality [1,2]. They provide a link between the physical world and virtual reality [3]. The Digital Twin paradigm includes, among other techniques, both Reduced Order Modelling [4,5] and Machine Learning [6]. All these techniques are based on mathematical models which allow building up a virtual reality environment able to simulate the real world in real time. Digital Twins are very useful in Industry 4.0 for industrial design and system control and/or when a precise manufacturing is required.

Reduced order modelling is a numerical strategy aiming to transform complex, multi-variable computationally expensive simulation models, or datasets, into significantly less complex mathematical functions, through which describe and hence predict the system's behaviour, while preserving its main characteristics, see for example [7]. TWINKLE can work successfully on dense, sparse or unstructured data (see Fig. 1(a), (b) and (c) respectively). ROMs are the modern equivalent of classical charts, e.g. thermodynamic steam tables, in which the value of a quantity for a given set of the input parameters can be quickly looked up and determined since all solutions of the problem would have been previously computed and transferred to the chart. ROMs work by reducing the full scope of model functions to a much smaller set that encapsulate most of the systems fundamental dynamics. Furthermore, ROMs can be embedded in complex systems' model construction, where the system in study is divided in simpler blocks, having computed individual ROMs

* Corresponding author.
E-mail address: sizquierdo@itainnova.es (S. Izquierdo).

https://doi.org/10.1016/j.softx.2020.100419

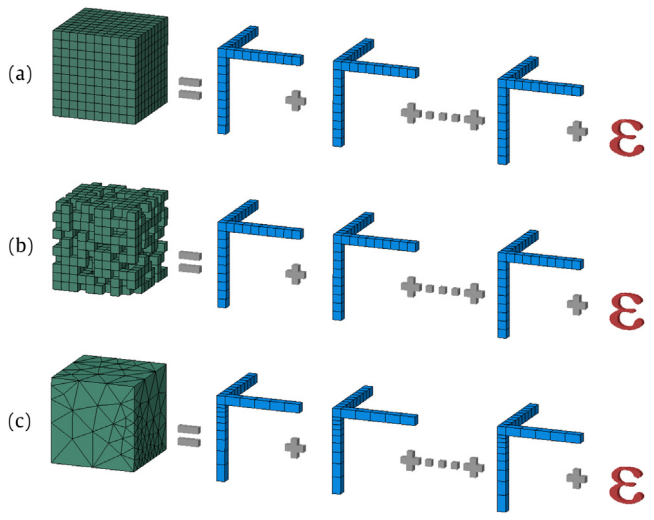

Fig. 1. Graphical visualization of model order reduction approaches for dense (a), sparse (b) and unstructured data (c).

on each of them [8]. ROMs techniques can be categorized into two main groups, as it follows.

- Intrusive methods [9–12].
- Non-intrusive methods [13–16].

Intrusive methods are typically used when the system's equations are known but their solution is not trivial. In this case it is possible to perform model order reduction to directly modify the system's equations. Such methods include Petrov–Galerkin projections, reduced basis method, Krylov subspaces, etc. Non-intrusive methods, on the other hand, can be either used when the equations of the system are well known or when they are completely unknown. These include approaches such as Proper Generalized Decomposition (PGD), Singular Value Decomposition (SVD) or Tucker decomposition. Such approaches do not affect system's equations, but they are performed directly on data. In this case data can be obtained through experiments, simulations or by mathematically solving equations, if these are known. If data are chosen to be acquired through experiments or simulations, a careful Design of Experiment (DoE) must be performed beforehand. In the latter case it is indeed crucial to achieve a deep understanding of the data in order to be able to discard unwanted scenarios and get a better output in the reduced order modelling. If data are not analysed correctly one may get unrealistic and biased results which will be therefore unable to predict the real phenomenon under investigation. In the worse case scenario a bad data interpretation could even lead to the computation of a ROM unable to adjust to the available data. It is therefore very important not only to perform a proper DoE, but also to conduct an efficient and accurate pre-process data analysis.

2. Software description

A complex system is usually described by the interaction of different parameters. Let us assume one would like to evaluate an output variable. This will depend on many other variables (inputs) of the system. Moreover, some system's variables will not affect the output only, but they will appear to be intertwined. Therefore obtaining a mathematical relation between the output and all the inputs of a system seems to be a pretty tough task, especially when, as often occurs, the input variables are interconnected. The possibility to describe the system's output by means of independent functions, one for each variable separately and related to the output variable is exactly the basic idea behind the mathematical implementation of this library. The approach is based on the assumption that a problem of N, not necessarily independent, variables can be rewritten as the product of N one-dimensional functions, one for each of the variables of the system, as shown in Eq. (1):

$$F(v_1, \ldots, v_N) = \sum_{m=1}^{M} \alpha_m \prod_{n=1}^{N} f_{m,n}(v_n) \tag{1}$$

where M is the order of approximation of the reduced order model and $\alpha_m, m = 1, \ldots, M$, are weighting coefficients. The functions $f_{m,n}$, in their most simple form, are piecewise linear functions; hence the adjustment parameters are the positions and the values at which the functions change slope. The adjustment of all these parameters is carried out through a least square optimization.

The first term of the sum in Eq. (1) represents a first approximation of the system, being its corresponding coefficient the largest one, while the following terms would be corrections to it and will generally have lower coefficient values unless the correction only applies to a specific outliers population and does not affect the general trend of data. The implementation of tensor decomposition adopted by TWINKLE is based on [13] and [16].

2.1. Software architecture

An executable example, *runTwinkle*, together with a set of data are provided with the *Twinkle* library, for both *linux* and *win64* operative systems, although it is also possible to compile the uploaded code through the *g++* commands provided.

Through the executable file it is possible to use the library with the provided data, or external ones, by setting the different required parameters. For this purpose a *-help* guide can be displayed to support the user. TWINKLE library is composed of three main classes, as described below.

- **f1D.** It contains the one-dimensional function evaluation at different discretization points, see $f_{m,n}(v_n)$ in Eq. (1).
- **Term.** It contains the f1D information for each of the system's inputs and the corresponding term's weighting coefficient α_m in Eq. (1).
- **Twinkle.** It contains the Term information for all the computed terms of the ROM so that Eq. (1) can be computed. Through this class the user is able to perform several actions, as described in the Section below.

2.2. Software functionalities

TWINKLE's main functionalities, available within the class Twinkle as described in the previous Section, can be divided into two main subgroups as it follows.

- Calculate the ROM
  - Set the data on which the calculation is to be performed. The data format can be directly a matrix or a file, namely a *.csv* or *.txt*.
  - Set computational parameters, i.e. global and term tolerances, number of Alternate Least Squares (ALS) iterations for shape functions convergence calculation, number of shape function initialization for new convergence attempts and the numerical precision of the results.
  - Create a discretization net as shown in Fig. 2. It is possible to generate the discretization in several different ways: setting a unique number of discretization points

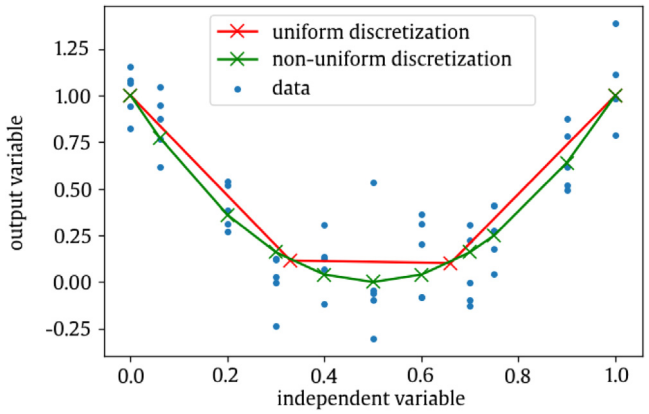

Fig. 2. Example of functions evaluated on two different types of discretization: uniform (red line), exact (green line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

equal for all dimensions (one-dimensional functions); passing a vector of discretization values; passing a file containing the discretization values (especially useful when the number of dimensions is big); passing a file containing specific values of the independent variable to be used for the generation of the discretization net (especially useful when a non-uniform discretization net is needed); generating an exact discretization, i.e. all the values taken from the variable are used to create the discretization net, in this case the discretization net is not necessarily uniform (see Fig. 2).

- Compute the ROM from a data file using Eq. (1). The user can set the number of terms and the shape functions initialization type, i.e. random or lineal. For each term a convergence tolerance must be fulfilled so that the overall algorithm convergence is tested.
- Write down the ROM's results, i.e. number of terms, discretization net, shape functions evaluations and weighting coefficients values, in a *.txt* file.
- Write down the ROM's prediction results for the original dataset.

- Evaluate the ROM
  - Start an evaluation mode instance of Twinkle by passing a previously calculated ROM's results file.
  - Write down the prediction results for the ROM evaluation, namely a *.txt* file.

- *General functionalities*
  - Write down the ROM's log.

In Fig. 3 an overview of the library is shown, where dashed lines indicate possible future implementations. No data preprocessing has yet been provided with the library mainly because there exist several different strategies and tools to implement an effective pre-processing, this way the user would not be obliged to follow any specific procedure when using TWINKLE. On the other hand future implementations related to data regularization are ongoing; this is planned to be achieved through different shape functions definitions. Furthermore an external loop could be defined in such a way that the algorithm would be able to discern whether to compute the ROM over the whole data manifold or perform a semi-automated data clusterization, defining hence a new scalable strategy.

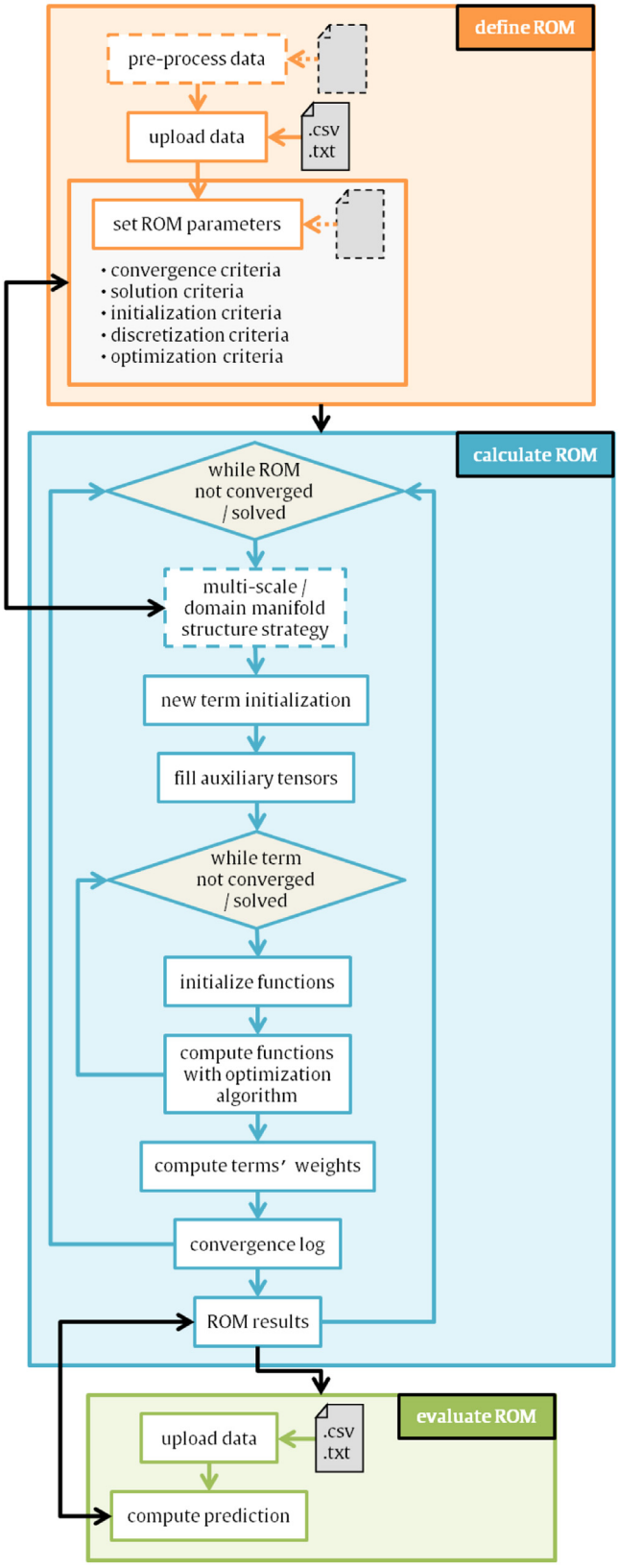

Fig. 3. Graphical visualization of TWINKLE library's present and future functionalities (dashed lines).

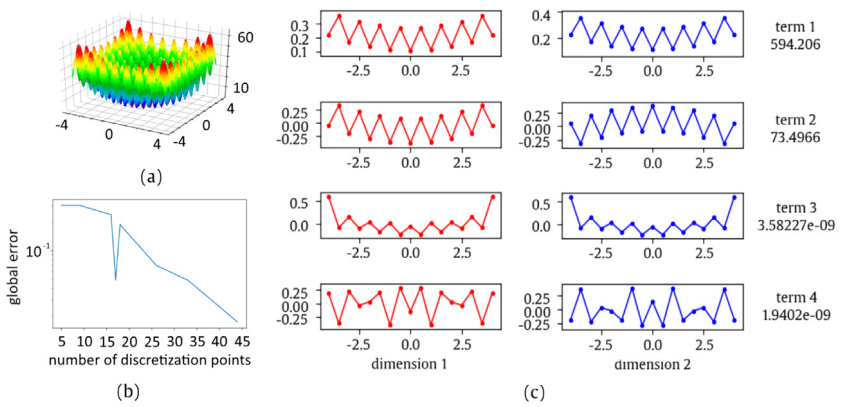

Fig. 4. Rastrigin function 3D plot (a), ROM results for Rastrigin 3D function (c), error in the prediction of the 3D Rastrigin function using increasingly refined discretization nets (b).

3. Illustrative examples

TWINKLE library is not linked to any specific field and its implementation makes it applicable to several different problems. Three illustrative simple examples for its usage are illustrated in the following Subsections.

3.1. Example 1: Rastrigin function retrieval

Approximations of analytical functions can be computed via ROM, where the function's values are passed as parameters to the model. An example of it has been obtained for the Rastrigin function Eq. (2), shown in Fig. 4(a):

$$f(x_1, \ldots, x_n) = An + \sum_{i=1}^{n} [x_i^2 - A\cos(2\pi x_i)] \tag{2}$$

where $A = 10, n = 2$ and $x_i \in [-4, 4]$ have been set for the calculation.

For sake of simplicity a Rastrigin function of two variables has been considered in this example, although TWINKLE is not limited to any given amount of input variables. The mathematical approximation of the Rastrigin function (see Eq. (2)) obtained by TWINKLE presents a good correlation with the real function and the most relevant terms are able to reproduce the function shape with good precision as shown in Fig. 4(c). This is especially important in large multidimensional data-sets where the individual influence of each independent variable on the output is very difficult to extract. A further analysis of the computation's precision has been performed. In Fig. 4(b) it is shown the decreasing global error of the computation against increasing number of discretization net's points. The local minimum obtained for seventeen number of points is achieved due to the geometrical characteristics of the retrieved surface, where all maxima and minima of the Rastrigin function correspond exactly to the discretization net's points.

3.2. Example 2: Simulink integration of ROMs for complex systems description

The MATLAB Simulink simulation tool is often used when coming to describe complex systems and to design models of them. Complex models can hence be decomposed into several interconnected blocks. TWINKLE library can be handful in such a scenario, since each, or some, of the aforementioned blocks can be implemented by solving equations obtained from different ROM simulations.

3.3. Example 3: Response surface approximation for designing

The behaviour of systems' components is usually parametrized in terms of geometric and material variables, however the exact calculation of each parameters' combination might be computationally expensive. Using TWINKLE on a reduced number of simulations' results, the desired component characteristic, e.g. the structural resistance of a given part of the system, can be obtained through a simple mathematical formula, as seen in Eq. (1), allowing the optimization of the specific component's behaviour in an effective and efficient way.

4. Impact

As stated previously, TWINKLE library cannot be limited to a specific task, it is a versatile and adaptable software instead. Its use can be relevant in big data processing and analysis with little computational cost. The application fields of TWINKLE vary from mechanics, to fluid- or thermodynamics, as well as economics, statistics, biology, medicine and basically any field involving big amounts of data, aiming to solve complex multi-physics or more generally multi-scale problems preserving the original input variables in the output results, which makes its result easy to interpret. The scalable concept defined within the algorithm allows setting the basis for a new strategy that presents many similarities with the multi-levels neuronal net for deep learning, providing the possibility to model data by defining the underlying manifold space not necessarily using a single tensor but a topology of them to better describe the phenomenon's space when data correlation appears to be cluster-like.

5. Conclusions

TWINKLE library allows minimizing computational cost as well as gaining precision in ROM computation, especially in case of sparse/unstructured data. Furthermore, the software allows processing big datasets and it has been programmed in an extensible way so that future implementations are taken into account in the algorithm's structure. Moreover, thanks to its extensibility, many of the library's methods can be updated and/or overloaded without altering the main software's structure; it is hence possible to use TWINKLE for many different purposes and fields of study, from fluid-dynamics, to material science or medicine.

TWINKLE is based on CPD allowing preserving the physics behind the phenomena under investigation by keeping the input variables themselves in the final output result. Moreover, as stated previously, the algorithm structure allows processing correctly both sparse and unstructured data (see Fig. 1), which is

not foreseen in CPD, such as SVD, Principal Components Analysis (PCA), PARAllel FACtor analysis (PARAFAC) or CANonical DECOMPosition (CANDECOMP) [1,2].

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.

References

[1] Tao F, Qi Q, Wang L, Nee A. Digital twins and cyber-physical systems toward smart manufacturing and industry 4.0: Correlation and comparison. Engineering 2019;5(4):653–61.

[2] Tao F, Zhang H, Liu A, Nee AY. Digital twin in industry: state-of-the-art. IEEE Trans Ind Inf 2018;15(4):2405–15.

[3] Chinesta F, Cueto E, Abisset-Chavanne E, Duval J, Khaldi F. Virtual, digital and hybrid twins: A new paradigm in data-based engineering and engineered data. Arch Comput Methods Eng 2018. http://dx.doi.org/10.1007/s11831-018-9301-4.

[4] Le Clainche S, Vega JM. A review on reduced order modeling using dmd-based methods. In: IUTAM symposium on model order reduction of coupled systems, Stuttgart, Germany, May 22–25, 2018. Springer; 2020, p. 55–66.

[5] Mignolet MP, Przekop A, Rizzi SA, Spottswood SM. A review of indirect/non-intrusive reduced order modeling of nonlinear geometric structures. J Sound Vib 2013;332(10):2437–60.

[6] Wuest T, Weimer D, Irgens C, Thoben K-D. Machine learning in manufacturing: advantages, challenges, and applications. Prod Manuf Res 2016;4(1):23–45.

[7] Quarteroni A, Rozza G, et al. Reduced order methods for modeling and computational reduction, vol. 9. Springer; 2014.

[8] Ghnatios C, Masson F, Huerta A, Leygue A, Cueto E, Chinesta F. Proper generalized decomposition based dynamic data-driven control of thermal processes. Comput Methods Appl Mech Engrg 2012;213:29–41.

[9] Rowley CW, Colonius T, Murray RM. Model reduction for compressible flows using POD and Galerkin projection. Physica D 2004;189(1–2):115–29.

[10] Rozza G, Veroy K. On the stability of the reduced basis method for Stokes equations in parametrized domains. Comput Methods Appl Mech Engrg 2007;196(7):1244–60.

[11] Baur U, Benner P, Feng L. Model order reduction for linear and nonlinear systems: a system-theoretic perspective. Arch Comput Methods Eng 2014;21(4):331–58.

[12] Chinesta F, Cueto E. PGD-based modeling of materials, structures and processes. Springer; 2014.

[13] El Halabi F, Gonzalez D, Chico-Roca A, Doblare M. Multiparametric response surface construction by means of proper generalized decomposition: An extension of the PARAFAC procedure. Comput Methods Appl Mech Engrg 2013;253:543–57.

[14] Cichocki A, Zdunek R, Phan AH, Amari S-i. Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation. John Wiley & Sons; 2009.

[15] Willcox K, Peraire J. Balanced model reduction via the proper orthogonal decomposition. AIAA J 2002;40(11):2323–30.

[16] Ibáñez R, Abisset-Chavanne E, Ammar A, González D, Cueto E, Huerta A, et al. A multidimensional data-driven sparse identification technique: The sparse proper generalized decomposition. Complexity 2018;2018.