Digital Twin Operational Platform for Connectivity and Accessibility using Flask Python

Matthew S. Bonney
Department of Mechanical Engineering
The University of Sheffield
Sheffield, UK
Email: m.bonney@sheffield.ac.uk

Marco de Angelis
Institute for Risk and Uncertainty
University of Liverpool
Liverpool, UK

David Wagg
Department of Mechanical Engineering
The University of Sheffield
Sheffield, UK

Mattia Dal Borgo
Department of Mechanical Engineering/ DMMS Lab
KU Leuven/ Flanders Make
Leuven, Belgium

***Abstract*—This paper demonstrates an example of an open-source, modular, and system independent framework that has been developed by a team of researchers working for the UK based DigiTwin project. The example in this paper is based on a framework for a *digital twin operational platform* (DTOP) that uses Flask Python, and is named DTOP-Cristallo. DTOP-Cristallo is an operational platform that uses HTML web pages as the interface between the digital twin simulations, both python based and 3rd party software based, and the user. This framework is particularly useful for connectivity with users, since it can deploy the digital twin with multiple accessibility options to accommodate a wide variety of intended uses.**

I. Motivation

The digital twin concept is a hot research topic, as described in the recent review papers — see for example [1], [2], [3], [4], [5], [6]. In the context of engineering applications, digital twins has four main elements (i) models (both physics- and data-based), (ii) data, (iii) digital connectivity, and (iv) representations of knowledge (e.g. contextual and expert knowledge) [7]. In order to realise a digital twin in practice, an *operational platform* is required. This will need to incorporate the necessary software and hardware components that enables the digital twin and user to interact with the corresponding physical twin. In this paper we will refer to such a combination of software and hardware as a *digital twin operational platform* (DTOP). The idea of a DTOP is closely related to that of a digital twin information system which is a concept that has been developed in the construction industry [8].

In terms of digital twin software developments to date, most activity has been driven by proprietary software vendors (see review in [4]) resulting primarily in closed-source products. However, there are major benefits by enabling interoperability between different parts of the digital twin and even between multiple different twins. Therefore an objective of the research community is to create greater openness in platforms and the associated software being used, and this is already being pursued in the area of Internet-of-Things (IoT) — see for example [9].

This ethos is applied to the development of an open software that consists of a prototype web-based digital twin operational platform called DTOP-Cristallo. The platform has been developed using the Python/Flask framework in order to provide a user interface via web pages. This format should maximize accessibility for as many users as possible. It also allows a direct route for connectivity to the physical twin and other cloud-based services that may be required. However, there are trade-offs to be made. For example, cyber security of the digital twin is much more difficult to manage using a web-based platform.

The accessibility options for Flask are based on how the user connects to the Flask server. In general, there are three main categories of this connections as pictured in Figure 1. The first category is a standalone connection similar to the software developed by proprietary vendors. This allows a single user/machine access to the digital twin. Using a standalone DTOP gives the highest level of inherent security, but severely limits any collaborative or multi-discipline work to be performed on the system. On the counterpoint, the most accessible category is a server or web-based DTOP. This allows for a multitude of users access to the digital twin, however considerations such as security must be taken into consideration. Additionally, this allows for easy cloud-based resources such as virtual computing and databases. As a compromise between these two extremes, the final category is a LAN-based DTOP. This allows higher security of being locally on site, access local high-performance computing, and collaborative efforts by allowing multiple users simultaneously.

II. Framework

Following the general definition of the envisaged digital twin operational platform, in this Section we briefly outline how the implementation of the prototype DTOP-Cristallo fits within the general open-source framework. The language at the core of the DTOP-Cristallo is Python, which is licensed under the *PFS license agreement*. In addition to the PFS license,

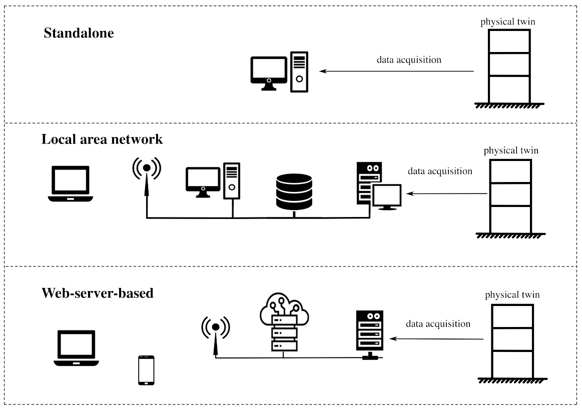


Fig. 1. Connections Categories for Flask

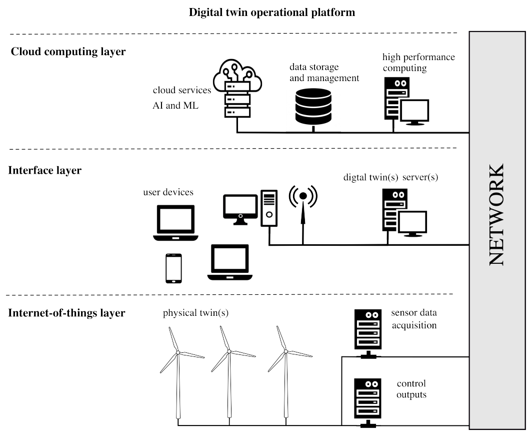


Fig. 2. Categorisation of the Layers of a DTOP

starting from Python 3.8.6, examples, recipes, and other code in the documentation are dual licensed under the PSF and the *Zero-Clause BSD license*. The choice of Python as the core language is because it is to date a dominant programming language in a variety of web-based and scientific applications. Despite the criticism about its speed, Python can also be used to run expensive algorithms thanks to the widely available and accessible libraries for vectorized computing like Numpy, and high-performance computing like CUDA.

Within the Python ecosystem, Flask stands out as a server tool for a variety of reasons. First, Flask is computationally light-weight, is released under the *BSD 2 license agreement*, and has no auxiliary requirements for accessing various aspects such as databases, validating web forms, authenticating users or other high-level tasks. This is in contrast to other platform generation tools, where most choices are hard to change or adapt to specific needs. Flask has three main dependencies:

- routing, debugging and Web Server Gateway Interface (WSGI)
- Jinja2 template
- command-line integration with Click

Because these dependencies come with the installation of Flask, the only requirement to run Flask is a computer with Python installed. Flask has also been chosen because of its accessibility and its thorough documentation, brought to life by the popular blog of Miguel Grinberg [10].

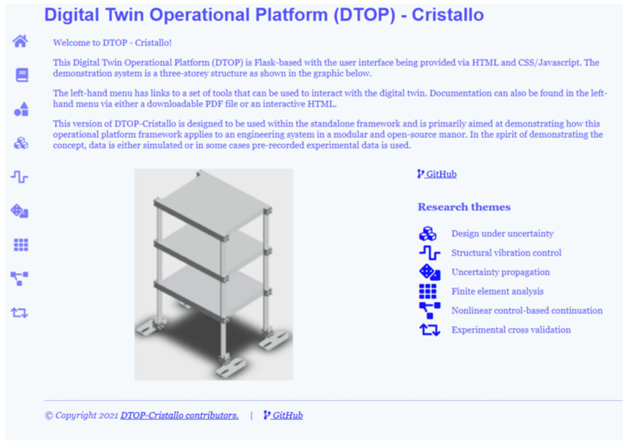
Flask provides the flexibility to structure the code in a way that the web-interface design is fairly independent from the development of the underlying mathematical code. This underlying mathematical code includes the code for dispatching the information gathered from the user, recorded data, and the pure mathematical simulations, leading to the utilisation of high-performance computing. The design of the web-interface can be done using the popular triad HTML5, CSS, and Javascript for responsive, interactive and animated browser-based graphical interfaces.

The open-source Plotly graphing libraries, under the MIT licence, were used to make the interactive plots. There are also working examples of Flask connecting with other popular tools for graphical interfacing, like React.js, which opens up exciting possibilities for non-browser-based applications. With Flask, the popular dichotomy frontend/backend can be fully exercised with only a few lines of code. Using Flask, the power of the high-level Python programming language can be fully harnessed, providing the inter-connectivity for multiple micro components that range from scientific computing to data management up to graphical interfacing. With this setup, the broad user-base can use a browser to interact with multiple components of the digital twin and trigger different computations, without any particular programming knowledge, whilst the knowledgeable user can download the open-source project and make the necessary changes. It is the hope that this setup will make digital twin technology more widely and easily accessible by the engineering community.

To aid in the development of this framework, a DTOP is separated into three levels as pictured in Figure 2. The first layer is the interface layer. This is the layer that contains aspects such as Flask, HTML, and the scientific simulations. DTOP-Cristallo has been focusing on the development of this layer with future work to incorporate and develop the other layers. The other two layers are focused on the software and hardware aspects of a DTOP. For the hardware aspects, the IoT layer focuses on the connection between the digital and physical twins. This includes the transfer of data from the physical to digital twin via the sensors and the transfer of data from the user to the physical twin. The last layer is the cloud computing layer. This involves the connection of the digital twin to remote computing options such as high-performance computing, virtual computing, and database storage.

III. Prototype Platform DTOP-Cristallo

DTOP-Cristallo is currently comprised of six individual tools/simulations that are used for various engineering applications. These tools range from nonlinear dynamic simulations, to uncertainty quantification, to comparing experimental and numerical data. To access these tools, links are provided on DTOP-Cristallo's home page and the side bar on each tool.



Digital Twin Operational Platform (DTOP) - Cristallo

Welcome to DTOP - Cristallo!

This Digital Twin Operational Platform (DTOP) is Flask-based with the user interface being provided via HTML and CSS/Javascript. The demonstration system is a three-storey structure as shown in the graphic below.

The left-hand menu has links to a set of tools that can be used to interact with the digital twin. Documentation can also be found in the left-hand menu via either a downloadable PDF file or an interactive HTML.

This version of DTOP-Cristallo is designed to be used within the standalone framework and is primarily aimed at demonstrating how this operational platform framework applies to an engineering system in a modular and open-source manor. In the spirit of demonstrating the concept, data is either simulated or in some cases pre-recorded experimental data is used.

GitHub

Research themes

- Design under uncertainty
- Structural vibration control
- Uncertainty propagation
- Finite element analysis
- Nonlinear control-based continuation
- Experimental cross validation

© Copyright 2021 DTOP-Cristallo contributors. | GitHub

Fig. 3. DTOP-Cristallo's Home Page

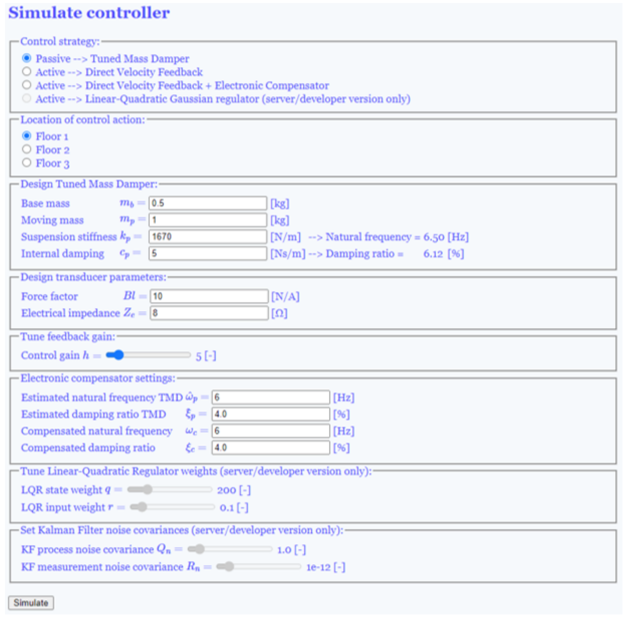
This can be seen in Figure 3, which is the main home page for DTOP-Cristallo. To discuss the programmed tools, they are classified into three main categories, self-contained, file IO, and 3rd party simulations.

A. Self-Contained Tools

Out of the six tools currently programmed, three of them are fully contained in within the python programming. These are, Design under uncertainty, Structural vibration control, and Uncertainty propagation. While each is programmed by different programmers with different styles, all three of these only require the functions specifically written by the programmer or built-in for python. For simplicity, these tools are split into two separate files: one that contains the scientific code, or functions, that perform the main calculations through the input by the user and the other one that interacts with Flask, gathers the user's input, call the main calculation, and sends the results back to Flask. This separation is particularly useful for the development of DTOP-Cristallo because it minimizes the required knowledge for a new programmer to have in order to contribute with their scientific code.

To demonstrate how the user interacts with DTOP-Cristallo and the programmed tools, Figure 4 shows the input selection options used in the structural vibration control tool. This tool investigates three distinct vibration control algorithms that can be used to reduce the resonance response of a mechanical structure. In general, this is very important in order to reduce the large oscillations that the structure can experience during an external harmonic excitation such as an earthquake or wind-based loading. These various algorithms include both passive and active methods to achieve this result. The user can apply various algorithm parameters in order to determine an optimal set of parameters.

This tool shows how a user interacts and gives parameter values to the simulation. Since the interface is written in HTML, the input can be a variety of values. For this specific tool, there are Boolean input, text-based numerical values, and numerical sliders. Some other possible inputs include text entries and file locations. Once the user has made their selection, the information is sent to Flask and the tool upon clicking the "Simulate" button below the parameter boxes.



Simulate controller

Control strategy:
- ● Passive --> Tuned Mass Damper
- ○ Active --> Direct Velocity Feedback
- ○ Active --> Direct Velocity Feedback + Electronic Compensator
- ○ Active --> Linear-Quadratic Gaussian regulator (server/developer version only)

Location of control action:
- ● Floor 1
- ○ Floor 2
- ○ Floor 3

Design Tuned Mass Damper:

Base mass	$m_b =$	0.5	[kg]	
Moving mass	$m_p =$	1	[kg]	
Suspension stiffness	$k_p =$	1670	[N/m]	--> Natural frequency = 6.50 [Hz]
Internal damping	$c_p =$	5	[Ns/m]	--> Damping ratio = 6.12 [%]

Design transducer parameters:

Force factor	$Bl =$	10	[N/A]
Electrical impedance	$Z_e =$	8	[Ω]

Tune feedback gain:

Control gain $h =$ 5 [-]

Electronic compensator settings:

Estimated natural frequency TMD	$\hat{\omega}_p =$	6	[Hz]
Estimated damping ratio TMD	$\xi_p =$	4.0	[%]
Compensated natural frequency	$\omega_c =$	6	[Hz]
Compensated damping ratio	$\xi_c =$	4.0	[%]

Tune Linear-Quadratic Regulator weights (server/developer version only):

LQR state weight $q =$ 200 [-]
LQR input weight $r =$ 0.1 [-]

Set Kalman Filter noise covariances (server/developer version only):

KF process noise covariance $Q_n =$ 1.0 [-]
KF measurement noise covariance $R_n =$ 1e-12 [-]

Simulate

Fig. 4. User's Input Options for Structural Vibration Control Tool

B. File IO Tools

The second category of tools relate to the use of external data through data-files that store information. There are two tools in DTOP-Cristallo that fall under this category, Nonlinear control-based continuation and Experimental cross-validation. These are categorised by the use of an external file that is maintained within the Git repository. For the experimental cross-validation, both previously calculated numerical model and experimental results are stored as numpy arrays. This tool gathers the information from the arrays and plots them against each other. An example of this information is seen in Figure 5. Currently, this tool purely plots the comparison between the prototypes and their numerical models. However, this tool can be expanded to test and quantify a few different important aspects. One of the most important aspects that it can quantify is the part-to-part variability that is inherent from manufacturing tolerances and material variability. Since four nominally identical prototypes were constructed at separate locations using different material stock, the dynamic characteristics of this structures are different from each other. While this is expected, keeping the variability to acceptable levels is important to ensure the system's robustness.

The nonlinear control-based continuation uses the files in a more interactive method by both reading and writing to the saved numpy arrays. This tools consists of two frequency sweeps (both increasing and decreasing) and a numerical continuation on the nonlinear properties of the structure. To perform the calculations, each evaluation appends its value to the numpy array that is saved in the file. While this method of

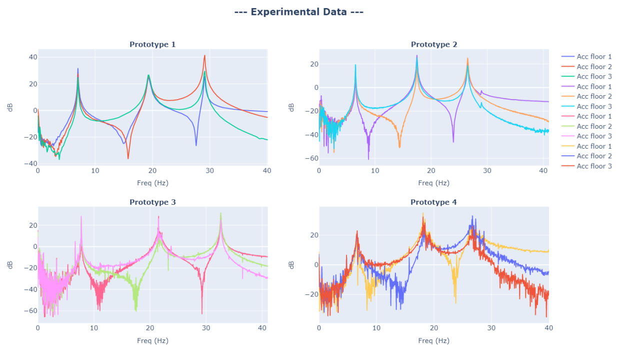


Fig. 5. Example Output of Experimental Cross-Validation Tool

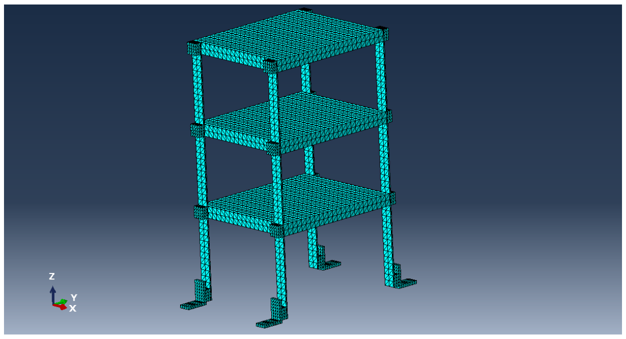


Fig. 6. Default Mesh Used in the Finite Element Analysis Tool

programming is not optimal, it does showcase an interesting implementation that can be utilised for other systems.

C. 3rd Party Simulations Based Tools

The last category currently implemented into DTOP-Cristallo is based on the connectivity to external programs. There is currently one tool that fits within this category, which is finite element analysis. This tool takes in user parameters such as, material properties, analysis parameters, and mesh information, then generates an ABAQUS run script that can perform the finite element analysis. The system with the default mesh size can be seen in Figure 6. Finite element analysis is a very commonly used simulation in structural design. This tool is particular because it is able to modify the material properties and predict the dynamic characteristics of the structure. There are two main uses of this tool with the first being a model calibration procedure. This would incorporate experimental data and modify the material properties to accurately model the physical twin. The other use is for investigating a possible material/heat treatment change.

In order to perform this analysis, the tool takes some required data (geometry data, material assignment, etc.) from a text file, then combines that information with the user provided values to generate a python-based run script. ABAQUS, since it is based in python, can take this run script and perform the finite element analysis. This generates multiple files, but with two main files of interest are the output database $*.odb$ and the text-based report $*.dat$. Both of these files contain the main simulation results, namely the natural frequencies, for the simulation. This can also be partially classified as a file IO tools since it uses verbose text files to create a python run script file. The reason for this separate category is the fact that DTOP-Cristallo also calls ABAQUS to run the analysis.

IV. Remarks

DTOP-Cristallo `https://github.com/Digital-Twin-Operational-Platform/Cristallo` uses a variety of tools that is commonly used in structural engineering design. This is developed as a system-based example of the uses of a DTOP. With this use, DTOP-Cristallo mimics a portion of what is commonly sold in the industry as a digital twin, but in an open-source and modular nature compared to the proprietary and bespoke software sold. A DTOP can also be used for a research project, but is not demonstrated within DTOP-Cristallo. In a similar DTOP, it is being used for designing experimental testing. This connects the results from an experiment that automatically get stored in a database, performs an analysis, then gives the experimental conditions for the proceeding experiment. For this system, the DTOP determines the next environmental temperature used to create an accurate temperature-dependent surrogate model using the minimum number of experimental results within a specified budget.

DTOP-Cristallo is a novel tool that demonstrates an open-source and modular framework for using an operational platform to implement and access a digital twin. It contains a selection of commonly used simulations/analyses tools for a variety of structural dynamic systems. Flask is used for DTOP-Cristallo to provide a flexible interface for accessing the digital twin and its tools. DTOP-Cristallo gives access to the digital twin of a scaled three-storey structure and is able to perform a variety of simulations and give access to experimental data.

Current work in the development of DTOP-Cristallo is focused on two main aspects. The first aspect is the implementation and study of the connectivity between the digital and physical twins in an open-source framework. This is mainly comprised of two parts, first is the reading of sensors using the DTOP and secondly is the sending of information to the physical twin for situations such as manual earthquake mitigation or shutdown. The second aspect is centered around the development of a server-based version. This gives access of the digital twin to the approved users from any location and device that has internet access. Having this increases in accessibility greatly expands the usability of digital twins for remote systems and international projects.

Acknowledgement

The authors would like to acknowledge the support of EPSRC via grant number EP/R006768/1. All the code and data presented in this paper can be accessed at the following GitHub repository: https://github.com/Digital-Twin-Operational-Platform/Cristallo

References

[1] A. Fuller, Z. Fan, C. Day, and C. Barlow, “Digital twin: Enabling technologies, challenges and open research,” *IEEE Access*, 2020.

[2] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, “Characterising the digital twin: A systematic literature review,” *CIRP Journal of Manufacturing Science and Technology*, 2020.

[3] M. Liu, S. Fang, H. Dong, and C. Xu, “Review of digital twin about concepts, technologies, and industrial applications,” *Journal of Manufacturing Systems*, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0278612520301072

[4] R. Minerva, G. M. Lee, and N. Crespi, “Digital twin in the iot context: a survey on technical features, scenarios, and architectural models,” *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1785–1824, 2020.

[5] D. J. Wagg, K. Worden, R. J. Barthorpe, and P. Gardner, “Digital Twins: State-of-the-Art and Future Directions for Modeling and Simulation in Engineering Dynamics Applications,” *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems Part B Mechanical Engineering*, vol. 6, no. 3, 05 2020, 030901. [Online]. Available: https://doi.org/10.1115/1.4046739

[6] S. A. Niederer, M. S. Sacks, M. Girolami, and K. Willcox, “Scaling digital twins from the artisanal to the industrial,” *Nature Computational Science*, vol. 1, no. 5, pp. 313–320, 2021.

[7] P. Gardner, M. Dal Borgo, V. Ruffini, A. J. Hughes, Y. Zhu, and D. J. Wagg, “Towards the development of an operational digital twin,” *Vibration*, vol. 3, no. 3, pp. 235–265, 2020.

[8] R. Sacks, I. Brilakis, E. Pikas, H. S. Xie, and M. Girolami, “Construction with digital twin information systems,” *Data-Centric Engineering*, vol. 1, 2020.

[9] M. Platenius-Mohr, S. Malakuti, S. Grüner, J. Schmitt, and T. Goldschmidt, “File-and api-based interoperability of digital twins by model transformation: An iiot case study using asset administration shell,” *Future Generation Computer Systems*, vol. 113, pp. 94–105, 2020.

[10] M. Grinberg, *Flask web development: developing web applications with python*. ” O’Reilly Media, Inc.”, 2018.