Companion code and data for the calibration of a UAV digital twin, as presented in “A Probabilistic Graphical Model Foundation for Enabling Predictive Digital Twins at Scale”

Table of Contents

- Overview
- System Requirements
- Instructions
- Contact

Overview

This code is a companion to an academic research paper. If you use this work in an academic context, please cite the following publication(s):

Kapteyn, Michael G., Jacob V.R. Pretorius, and Karen E. Willcox. **A Probabilistic Graphical Model Foundation for Enabling Predictive Digital Twins at Scale.** arXiv preprint arXiv:2012.05841 (2020). https://arxiv.org/abs/2012.05841

```
@article{kapteyn2020probabilistic,
  title={A Probabilistic Graphical Model Foundation for Enabling Predictive Digital Twins at Scale},
  author={Kapteyn, Michael G and Pretorius, Jacob VR and Willcox, Karen E},
  journal={arXiv preprint arXiv:2012.05841},
  year={2020}
}
```


Keywords: UAV, digital twin, graphical model, Dynamic Bayesian Network, experimental calibration

This folder contains code and data to reproduce the results in the associated paper. In particular, the provided scripts will reproduce all graphs in Figure 5 and Table 1.

Calibration step 1

We provide geometry data (`datafiles/step1/geometry_data.csv`) and code (`calibrationStep1.mlx`) to perform the update on geometric parameter estimates. This recreates Table 1, columns 1-3.

Calibration step 2

We provide measured load-displacement data (`datafiles/step2/load_displacement_measurements.csv`), prior model predictions of load-displacement data (`datafiles/step2/load_displacement_model.csv`) and code (`calibrationStep2.mlx`) to perform an update on the young's modulus scaling parameter, e. This recreates Figure 5, and Table 1, column 4.

Calibration step 3

This step is broken into 3 parts:

Step 3_1: Data processing

We provide experimental strain vs. time data (`datafiles/step3/initialconditiondata_X.DAT`) for the initial condition experiments and code (`calibrationStep3_1_processdata.mlx`) to post-process this data and extract modal natural frequencies and damping ratios for the first two bending modes.

Step 3_2: Optimize point masses

This step requires evaluating the computational digital twin model in an optimization loop in order to fit point-masses to the structure which match the natural frequencies and damping ratios computed in Step 3_1. We provide Python code (`calibrationStep3_2_optimize_point_masses.py`) which was used to run this optimization on our UAV structural model. The structural analysis model used to generate the paper's results is Akselos Integra v4.5.9. Since the Akselos Integra software is proprietary and was used under license, we are unable to provide its source code. As a result **the provided Python code will not run in the provided state**. The provided Python code includes references to three (self-explanatory) functions that will need to be filled in for your choice of structural model:

`modify_material_properties_in_structural_FEA_model(Emultiplier)`

`modify_point_masses_in_structural_FEA_model(point_mass_dict)`

`frequencies = run_structural_FEA_model_and_return_natural_frequencies()`

We provide the output obtained from running this script on our UAV structural model in `datafiles/step2/massoptimizationresults.csv` .

Step 3_3: Process Optimization results

We provide the script `calibrationStep3_3_processoptimizationresults.mlx` which reads in `massoptimizationresults.csv` and post-processes the results in order to produce posterior estimates for the parameters `m`, `alpha`, and `beta`. This reproduces Table 1, columns 5-7.

Requirements

.mlx files

The majority of provided code files are [MATLAB live code files](). These are a code notebook format which contains descriptive text alongside the code.

`.mlx` files are compatible with MATLAB version R2016a or later

In the folder `legacyformat` we also provide versions of these scripts that have been converted to `.m` files, which are compatible with any version of MATLAB and can also be viewed (but not exectuted) in a text editor.

The following functions used in the scripts require toolboxes to be installed:

- `fitdist`, `normpdf`, and `normrnd` are from the `Statistics and Machine Learning Toolbox`
- `decimate`, and `findpeaks` are from the `Signal Processing Toolbox`

The software has been tested on MATLAB R2020a and MATLAB R2018a.

.py file

As described [above](), this code will not run as-is. It requires the user to provide an interface with a structural model.

This code requires Python 3, and was tested using Python 3.7.4. The script leverages the following libraries:

- `numpy` (any version)
- `scipy` (any version containing `scipy.optimize.fmin`)

Instructions for use

Installation

This code requires no installation or compilation.

Execution

Simply open the relevant live script within MATLAB, set your working directory to the folder containing the script, then run the script.

If using the `legacyformat` code files, it is recommended that you use the [`publish`] functionality in MATLAB.

Expected output

The livescripts should run to completion without producing errors. Figures matching the data elements of Figure 5 and Table 1 in the paper should be generated throughout the scripts.

Each live script should take no more than a few minutes to run on a standard laptop or desktop machine.

Further Reading

Kapteyn, Michael G., Jacob VR Pretorius, and Karen E. Willcox. “A Probabilistic Graphical Model Foundation for Enabling Predictive Digital Twins at Scale.” arXiv:2012.05841 (2021). https://arxiv.org/abs/2012.05841

Contact

Michael Kapteyn mkapteyn@mit.edu