RaPId - A Parameter Estimation Toolbox for Modelica/FMI-Based Models Exploiting Global Optimization Methods

Meaghan Podlaski * **Luigi Vanfretti** * **Tetiana Bogodorova** *
Tin Rabuzin ** **Maxime Baudette** ***

* *Rensselaer Polytechnic Institute, NY, USA (e-mail: {podlam, vanfrl, bogodt2}@rpi.edu).*
** *KTH Royal Inst. of Tech., Sweden (email: rabuzin@kth.se).*
*** *Lawrence Berkley National Laboratory, CA, USA (email: baudette@lbl.gov).*

Abstract: This paper describes new additions to the Rapid Parameter Identification Toolbox (RaPId), which is an open-source MATLAB toolbox for parameter estimation using models developed with the Modelica language and exported with the functional mock-up interface (FMI) Standard. These additions include an updated graphical user interface (GUI), an optimization method utilizing multiple starting points for a gradient descent optimization, and examples for different cyber-physical system applications such as the Duffing-Holmes equation modeling in a form of electrical circuit and a hydroelectric power plant modeling.

Keywords: Dynamic models, Electric power systems, Global optimization, Parameter estimation, Software tools

1. INTRODUCTION

1.1 Motivation

In complex cyber-physical systems where testing opportunities are limited, such as power systems, reliable simulation based studies are necessary. Improved models can be created using parameter estimation methods from physical system data. The **Ra**pid **P**arameter **Id**entification toolbox (RaPId) was designed to carry out parameter identification on models using MATLAB/Simulink (Vanfretti et al., 2016). The toolbox was originally developed with modularity and extensibility in mind. Models developed using the equation-based, object-oriented modeling language, Modelica, could be coupled with the measurements and optimized using various gradient-based and heuristic optimization methods.

In this work, new optimization methods utilizing multi-start algorithms were added to the toolbox, as it was previously limited in obtaining accurate parameters for certain systems when the initial start point of the systems' parameters were far from global optimum. This toolbox is open-source and avaliable on Github at: `https://github.com/ALSETLab/RaPId` (ALSETLab (2020)).

⋆ This work was supported in part by the National Aeronautics and Space Administration under award number 80NSSC19M0125 as part of the Center for High-Efficiency Electrical Technologies for Aircraft (CHEETA), by Dominion Energy, by the New York State Energy Research and Development Authority (NYSERDA) under agreement numbers 137940 and 137951, by the Engineering Research Center Program of the National Science Foundation and the Department of Energy under Award EEC-1041877, and in part by the Center of Excellence for NEOM Research at King Abdullah University of Science and Technology.
The first author is supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1744655 and the Chateaubriand Fellowship of the Office for Science & Technology of the Embassy of France in the United States.

1.2 Related Works

Model validation tools have become increasingly important in analyzing highly complex cyber-physical systems, such as power systems. Many analysis tasks would benefit from a flexible parameter estimation tool like RaPId, such as identification of aggregate model parameters (Kalsi et al., 2011) and validation of overall grid dynamics (NASPI, 2015).

The original version of RaPId is described in Vanfretti et al. (2016), which outlines the software architecture for a MATLAB parameter estimation toolbox for functional mock-up units (FMU) (Modelica Association Project (2020)). The toolbox's goal is to offer a flexible solution that supports heterogeneous system models through exchange using standardized software methods, i.e. the Functional Mock-up Interface (FMI) standard, and enables easy addition of new optimization algorithms, simulation solvers, etc. The use of FMI allows users to import models from other modeling and simulation environments. Other MATLAB toolboxes are available for parameter estimation of continuous time models such as the SysId Toolbox (MathWorks (2020b)), CAPTAIN (Taylor et al. (2018)) and CONSTID (Garnier and Gilson (2018)), however they require that the models are defined exclusively in MATLAB/Simulink environment. The models used in RaPId can be directly imported from different tools, where models used in other toolboxes must be converted strictly to a linearized state-space model of a Simulink model prior to parameter estimation. RaPId aims to address this gap

and to provide certain automation features useful for the parameter estimation process.

1.3 Paper Contributions

This paper builds off of the existing RaPId MATLAB/Simulink toolbox to include the following new enhancements:

- Addition of `multiStart` as an option for optimization. It uses gradient based optimization methods (`fmincon`) for parameter estimation using multiple starting points.
- Updated and improved GUI to include the `multiStart` optimization processes.
- Parameter estimation for an electric circuit representation of a Duffing oscillator validated against measurements from a physical circuit. The data is generated from an electric circuit to identify the parameters of a model developed using Modelica. This provides another example for users who do not have power systems experience.
- Parameter estimation for a new power plant using phasor measurement (PMU) data added to RaPId.

2. SOFTWARE DESCRIPTION

The RaPId toolbox was originally developed to automate model validation, calibration, and parameter estimation for models available in Functional Mock-up Units (FMU) using model exchange variant from the Functional Mock-up Interface (FMI) standard Modelica Association Project (2020). FMUs can be generated from a variety of tools (see `https://fmi-standard.org/tools/`), and contain a casual representation of the original mode (e.g. it has specific inputs/outputs defining the causality). The resulting `*.fmu` is an archive file (i.e. a `*.zip` file) that contains the following: (i) an `*.xml` file with static model information, (ii) Source C/C++ code or a shared library (i.e. a `*.dll` in Windows OS) with the model equations converted to casual form and implementing the FMU Advanced Programming Interface, and (iii) additional resources (e.g. images, licenses, etc.). To import the FMU, RaPId uses a Simulink block or functions provided by the FMI Toolbox Modelon (2018).

The parameter identification process for the software is shown in Figure 1. RaPId takes reference time-series data along with simulation results to minimize a cost function and to determine an optimal set of parameters. The user defines parameters from the FMU to be calibrated, simulation and optimization solver to use, etc. The toolbox calibrates the model through an the iterative process shown in Figure 1:

(1) A reference measurement in the form of time series data is provided to the toolbox.
(2) The RaPId container applies algorithms for parameter identification, optimization method, and solver to simulate the FMU in Simulink.
(3) The FMI block defines the Modelica model exported as an FMU using model exchange in Simulink.
(4) The reference measurements and the simulation results are used to determine the error.
(5) RaPId uses the error to determine new parameters using the specified optimization method.

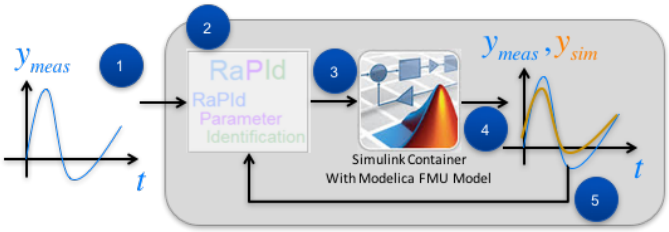

Fig. 1. RaPId parameter identification process.

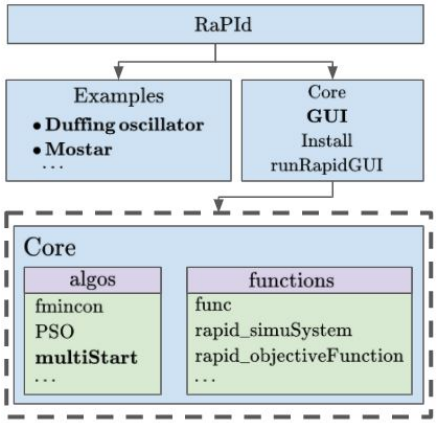

Fig. 2. RaPId software architecture implementation for parallel optimization, new additions to the toolbox are in bold.

2.1 Updated Software Architecture

The RaPId toolbox has the same architecture as described in (Vanfretti et al. (2016)). Figure 2 shows the relationship between each of the functions used in the toolbox. In the `core` folder of the toolbox, the file `rapid.m` manages all of the experiment settings such as the parameters to optimize, starting guess values, optimization bounds, and other simulation settings. The `rapid_objectiveFunction.m` file defines the cost function to determine the fitness of each parallel simulation result, user-defined functions can be added. The optimization is handled by a `multiStart` algorithm that can be found in the `algos` folder of RaPId.

The simulation of the models are configured to use FMI model exchange, so the solvers available in MATLAB/Simulink can be used. The FMUs can be implemented as FMU 1.0 and 2.0 by using the FMI Toolbox from (Modelon (2018)), which interfaces FMUs generated from Modelica tools with MATLAB/Simulink.

2.2 MultiStart Optimization

The capability to run objective functions with multiple starting points has been added to RaPId to improve parameter identification results. It relies on the framework outlined in (Ugray et al. (2007)). These algorithm files can be found in the `algos` folder in Figure 2. When the `parallel` option is selected, RaPId will use the `multiStart` solver from the Global Optimization Toolbox from MathWorks (2020a). This toolbox provides optimization functions for problems that contain multiple minima or maxima to find a global solution. This is useful for systems such as power grids, where parameter identification

process using grid measurements may result in identifying a result at a local optimum.

The optimization routine using multiple user-defined starting points and the `multiStart` solver method is shown in Algorithm 1. After all of the inputs and settings for the experiments are defined, multiple starting guesses for the parameters are given as inputs to the `multiStart` solver to find a global solution using the gradient based solver, `fmincon`.

Algorithm 1 Using MultiStart with gradient solver

Set up simulation experiment using RaPId:
p_i = i-th vector for all parameters to be calibrated
p_{min} = vector of lower limit of all parameters calibrated
p_{max} = vector of upper limit of all parameters calibrated
`model-info` = path to Simulink model and FMU, integration method.
t_{meas} = time vector of measurement data
y_{meas} = vector of measurement data
$cost\_function$ = cost function to calculate error
ϵ_0 = error tolerance
Set up FMU parameters, inputs, and outputs:
`parameterNames` = vector of names of parameters to be updated in FMU
`fmuInputNames` = vector of names of inputs to FMU (from measurements)
`fmuOutputNames` = vector of names of outputs from the FMU (to compare with measurements)
Set up the optimization algorithm in RaPId:
case = `'parallel'`
Run parameter estimation with RaPId:
(1) Using Multistart, assign one solver to each vector in x_i. Find solution using `fmincon`.
(2) Compare simulation results to measurements by computing error based off of the cost function defined during initialization. This determines the error, ϵ.
(3) If the error, ϵ, is below the error tolerance or the maximum number of iterations have been reached, the parameter set obtained is the final solution. If not, the parameters are then used as the input of the next RaPId iteration.

2.3 Updated RaPId GUI

The optimization routines and parameter identification of RaPId can be run from the command line or via a script saved as a `'*.m'` file or using a graphical interface. The GUI has been improved and updated to include the new parallel optimization algorithm settings, which is shown in Figure 3. The GUI has the option to load a pre-existing container in box A, which contains the experiment data outlined in Algorithm 1. This includes the FMU input, output, and parameter names, optimization starting guesses, and optimization bounds. It also contains the input data and reference data used to run the parameter identification (as visualized in Figure 1).

If a new container needs to be created, there is an option to load the FMU model path, input/output measurement data and names, and parameter names by clicking the 'ODE simulation settings' box in box B. The parameter limits and start values are defined under 'Simulink settings'. The drop down menu gives the options of multiple

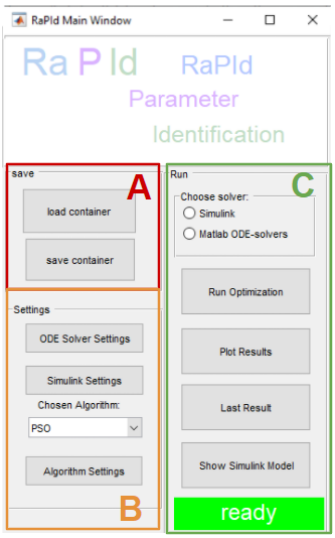

Fig. 3. Updated RaPId GUI

optimization methods to use with RaPId, where the 'parallel' setting is new and allows for inputs specific to the solver. The simulation settings can be saved in a container as a '.mat' file by pressing the 'Save Container' button. Once the problem and container have been properly configured, RaPId will start the optimization by pressing the 'Run Optimization' button. There are also options to plot the results from Simulink with the GUI, or results can be plotted by calling the variables from the command line.

3. UPDATED EXAMPLES

3.1 Model Development using Modelica and the FMI Standards

Modelica is an object-oriented, equation-based modeling language (Fritzson, 2004). These models are defined using differential-algebraic equations (DAEs). The connections to the model are not defined by input and output signals, but by acasual physical ports that couple the DAEs. Multiple engineering domains can be represented in the models, as the equation-based features allow for these aspects to be easily coupled together.

The models written in Modelica can be coupled with other simulation tools as an FMU through the FMI Standard (Modelica Association Project, 2020). There are two options for coupling models to other tools: model exchange and co-simulation. In model exchange, only the model is included in the FMU and the model uses the solver of the tool that is imported in for simulation (i.e. in the case of RaPId that is `MATLAB/Simulink`. Co-simulation includes the solver from the source of the FMU. RaPId only supports FMU import using model exchange.

3.2 Illustrative Electric Circuit Example

This version of RaPId now includes an example using an electric circuit of a Duffing oscillator to show flexibility and application of the tool outside of power and control system domains.

Electric Circuit Equivalent of the Duffing-Holmes Equation. The Duffing-Holmes equation (1) is a second order

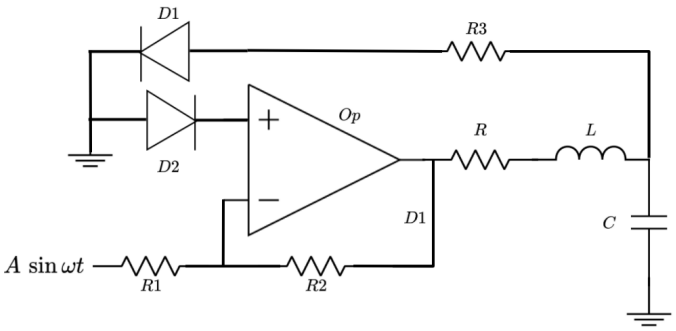

Fig. 4. Duffing-Holmes equation represented as an RLC oscillator circuit.

non-autonomous differential equation that can be represented physically as a second order analog circuit using an operational amplifier, allowing for the generation of chaotic waveforms (Tamaševiciute et al., 2008).

$$\ddot{x} + b\dot{x} - x + x^3 = a\sin\omega t \tag{1}$$

The electrical circuit is shown in Figure 4. It is an RLC oscillator with nonlinear elements in the positive feedback loop that results in choatic behavior. The Duffing-Holmes equation can be described using Kirchhoff's laws in Equations 2 and 3. The function $F_E(V_C)$ in Equations 3 and 4 is a nonlinear function of the capacitor voltage that Tamaševiciute et al. (2008) reduces into a piecewise function with three segments. In Equation 4, $k = R_1/R_2 + 1$, which is the gain of the amplifier and V^* is the bias voltage of the diode. Silicon diodes are used in this circuit, which have a voltage drop of approximately 0.5 V at 0.1 mA.

$$C\frac{dV_c}{dt} = I_L \tag{2}$$

$$L\frac{dI_L}{dt} = F_E(V_C) - I_L R + A\sin(\omega t - \pi) \tag{3}$$

$$F_E(V_C) = \begin{cases} -(V_C + kV^*), & V_C < -V^* \\ (k-1)V_C, & -V^* \le V_C \le V^* \\ -(V_c - kV^*), & V_C > V^* \end{cases} \tag{4}$$

Circuit Model and Physical Prototype. The circuit in Figure 4 is modeled in Dymola using the Modelica Standard Library (MSL) (Modelica Association, 2020). It is an open source Modelica library with components to model mechanical, electrical, thermal, fluid, and control systems. The model is exported as an FMU to use in RaPId within `MATLAB`/Simulink.

A physical electrical circuit was also constructed to record measurements to identify the true parameters of the RLC components in Figure 4. These measurements are recorded using the Analog Discovery 2, which is a USB oscilloscope and waveform generator (Digilent (2020)). This allows us to drive the physical circuit and model with the same sinusoidal input waveform.

Optimization Problem Setup. The parameter vectors, `parameterNames` and p_i, according to Algorithm 1 are the values of all resistances in the circuit. The upper and lower bounds have been determined based on the error tolerance of the physical components (e.g. a 10kΩ resistor has a 5% tolerance, so the realistic resistance of the component ranges from 9.5kΩ to 10.5kΩ). The measurement data is divided into a time vector, t_{meas}, and a measurement vector, $y_{meas} = [V_C, I_L]$. The output measurements have been selected to match the quantities needed to plot the chaotic behavior of the oscillator (Tamaševiciute et al. (2008)). In Equation (5), the lower and upper limits to the optimization problem are defined as p_{min} and p_{max}.

$$\min_x f(x) \text{ such that } \{\, p_{min} \le x \le p_{max} \,\}. \tag{5}$$

Using the `multiStart` optimization method with the gradient solver `fmincon`, the estimated parameter vector $\bar{p}$ is continuously optimized and updated to simulate the response of the system for each of the local solvers. The absolute difference between the simulation and measurements is calculated at each time step, as follows:

$$\epsilon_1 = \begin{bmatrix} V_c^{simulated} - V_c^{reference} \\ I_L^{simulated} - I_L^{reference} \end{bmatrix}^T \tag{6}$$

The objective function computed using the Frobenius norm from the mismatch ϵ_1. The sum of mismatches is calculated from the norms of the measurement/simulation pair at each time step, returning the fitness of the simulated model to the measurements for that parameter set. The fitness of each type of signal are weighted equally when optimizing the parameters by minimizing:

$$f(x) = \Sigma_{i=1}^{m}\Sigma_{j=1}^{n}(\epsilon_{ij} * \epsilon_{ij}) \tag{7}$$

Parameter Estimation Results. The parameter estimation was completed using three different optimization methods (`fmincon`, `multiStart`, and particle swarm optimization (PSO)) using measurement data collected from the circuit. The behavior that this model aims to replicate is shown in Figure 5; given the chaotic waveform, the results are shown in tabular form in Table 1 for clarity. The fmincon and PSO algorithms have an initial guess of [20.5, 10020, 10010, 9990]; the `multiStart` algorithm has an initial guess of [20.5, 10020, 10010, 9990; 19,10030, 10070,10000;20.1,11000, 10050,10300]. The stopping error tolerance is 1e-3 and 50 iterations.The `fmincon` method takes 12 iterations to solve with a residual of 1.82; PSO method takes 33 iterations with a residual of 2.24, and the `multiStart` method takes 17 iterations with a residual of 1.85.

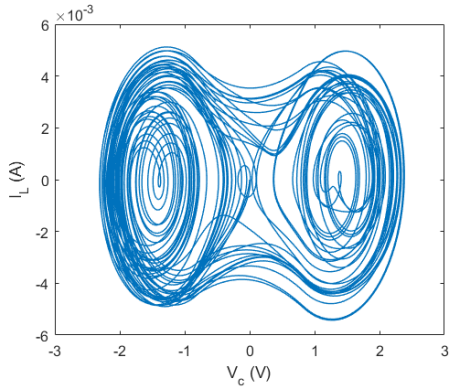

Fig. 5. Duffing-Holmes equation represented as a function of the capacitor voltage and the inductor current from the circuit in Figure 4.

Table 1. Resistance values and estimation results of the system in Figure 4.

Solver	R (Ω)	R1 (Ω)	R2 (Ω)	R3 (Ω)	Time(s)
Physical	20	10000	10000	10000	N/A
fmincon	20.56	10151.35	9998.06	10011.32	519
PSO	20.38	10114.07	10364.86	10036.86	1325
multiStart	19.9	10498.71	9999.38	9999.2	772

The multiStart is superior in computing parameter estimation in terms of computational time compared to the PSO solver. It takes computationally longer than the `fmincon` solver due to the fact that multiple starting points are evaluated.

3.3 Hydroelectric Power Plant Example

The parallelization algorithm is tested using phasor measurement data (PMU) obtained during the commissioning testing of a hydroelectric power plant. The model of the power system has previously been developed using the OpenIPSL Modelica library, which is an open-source power systems library that includes models verified against conventional power system software (Baudette et al. (2018)).

Power System Model. The model consists of three main components whose parameters need to be estimated: the generator, the automatic voltage regulator (AVR), and power system stabilizer (PSS). The system is configured as a single machine infinite bus (SMIB) system. The Modelica model is coupled to the SMIB line diagram is shown in Figure 6. The models used in this example are derived from the IEEE standard for power system excitation models IEEE (2020). The components are labeled as follows:

- A System data specifies the system's frequency and base power. The `machineData` block contains parameter data stored in a record, which is propagated to the corresponding system components. An additional sub-record with the results of every parameter calibration test is also included.
- B GENSAL generator model (with saliency effects).
- C ST5B AVR model (power electronic-based excitation system and Automatic Voltage Regulator (AVR)).
- D PSS2B PSS model (power system stabilizer).
- E Load, transmission line, and infinite bus connection.
- F FMU outputs, which are `Real` outputs, represent the simulated variables used in Eq.(8), i.e. `P`=$P_{out}^{simulated}$ and `Q`=$Q_{out}^{simulated}$.
- G Injected voltage disturbance to the voltage regulator.
- H Generator field current `XadIfd` with uniform noise.

The power system model is exported as an FMU using model exchange to optimize the parameters of the generator and the control system using phasor measurement unit (PMU) data. The model uses the change in machine voltage as an input. The outputs are the active and reactive power, terminal machine voltage, and field voltage of the generator, which are compared to the PMU measurements.

Optimization Problem Setup. The parameter vectors, parameterNames and p_i, according to Algorithm 1 are the generator impedances and control system gains and time constants. The upper and lower bounds, p_{min} and p_{max}, have been determined based on the physical limits of the machines from Kundur (1994). The measurement data

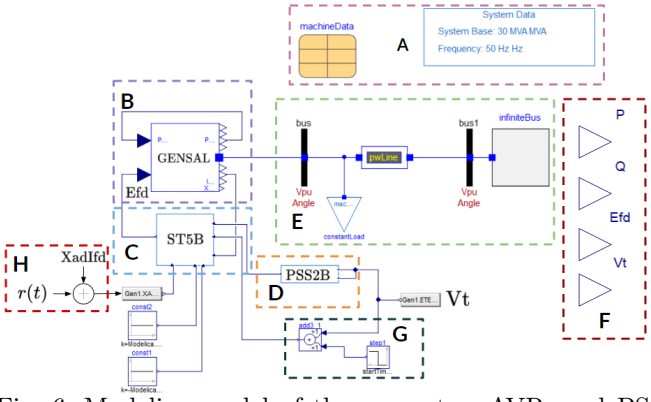

Fig. 6. Modelica model of the generator, AVR, and PSS connected to the rest of the system in Dymola.

is divided into a time vector, t_{meas}, and a measurement vector, $y_{meas} = [P_{meas}, Q_{meas}]$.

Using the `multiStart` optimization method with the gradient solver `fmincon`, the estimated parameter vector $\bar{p}$ is continuously optimized and updated to simulate the response of the system for each of the local solvers. The absolute difference between the simulation and measurements is calculated at each time step, as follows:

$$\epsilon_1 = \begin{bmatrix} P_{out}^{simulated} - P_{out}^{reference} \\ Q_{out}^{simulated} - Q_{out}^{reference} \end{bmatrix}^T \tag{8}$$

The objective function computed using the Frobenius norm in Equation (7) from the mismatch ϵ_1 between the simulation and PMU measurements from the generator's active and reactive power.

Parameter Estimation Results. The parameters are calculated using three different optimization methods: `fmincon`, particle swarm optimization, and `multiStart` algorithm. The `multiStart` solver is used with five starting points derived from the results of the sequential parameter updates from the results of the PSO optimization. The parameter bounds are defined from the machine parameters in Kundur (1994). These tests are run on three different computers [1] to compare the time to run the parameter estimation. All computers were configured with the Windows 10 OS and MATLAB 2020a. Table 2 shows the time elapsed to run the parameter estimation for each method. The EDGE solves the optimization problems on average 38.42 sec faster than the MSI GS65 and 20.67 sec faster than the Dell; this shows the value of parallelization as the EDGE has 12 cores while the MSI and Dell have 6 cores. Meanwhile, the optimization is faster on the Dell than the MSI with an average 17.75 sec, which is attributed to the higher base clock speed and 4 × larger RAM of the Dell.

The simulated results for each optimization method compared to the measurements are shown in Figure 7. This test is configured such that the active power stays constant throughout the simulation, and thus, the reactive power

[1] NextComputing EDGE XTa2 Creative Pro Workstation with an AMD Ryzen 9 3900x (12-core) CPU @3.8/4.6 GHz (base/max) and 128 GB of 36000 MHz DDR4 RAM, an MSI GS65 with an Intel® Core™ i7-9750H (6-core) CPU @2.60/4.50 GHz (base/turbo) and 32.0 GB of 2666 MHz DDR4 RAM, and a Dell Precision T3610 with Intel® Xeon®E5-1650 v2 (6-core) CPU @3.50/3.90 GHz (base/turbo) and 128 GB of 1866 MHz DDR3 RAM

Table 2. Optimization Solution Time

Machine	fmincon	PSO	MultiStart
EDGE XTa2	302.1489	453.4437	616.3450
Dell Precision T3610	311.2393	483.5012	630.4435
MSI GS65	329.1109	501.54	653.5334

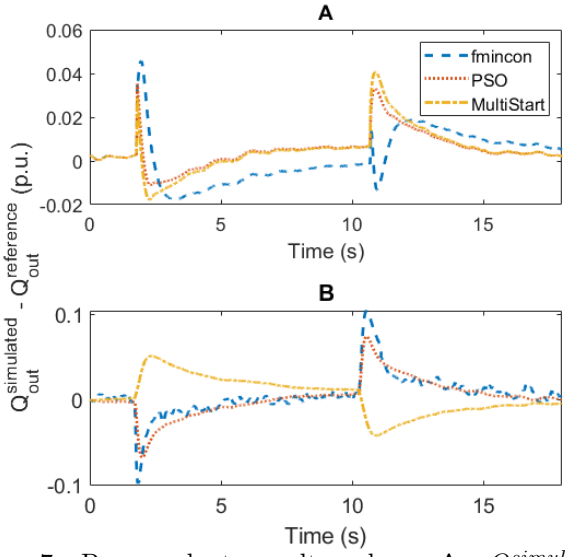

Fig. 7. Power plant results where **A:** $Q_{out}^{simulated} - Q_{out}^{reference}$, **B:** $Q_{out}^{simulated} - Q_{out}^{reference}$ when uniform standard noise with σ=0.05 is added to the signal `XADIFD` in block H in Fig. 6.

undergoes a 0.4 p.u. step. The `fmincon` and PSO results show how the model's response overshoots when the step is applied. The results obtained from the `multiStart` method are the best fit to the measurements.

4. CONCLUSION AND FUTURE DEVELOPMENTS

A new example that models the equivalent of Duffing-Holmes equation using the electrical circuit has been added to the RaPId toolbox, showing the tool's flexibility. It provides users that are not familiar with power systems a comprehensive example that is easy to understand.

An optimization using multiple starting points has been added to the RaPId toolbox, allowing for `multiStart` method for parameter estimation. This method has been tested for a hydroelectric power plant and its control system using commissioning test data. This optimization method produces better results than the previous particle swarm optimization (PSO) and constrained gradient optimization (`fmincon`) studied before. In the future, the RaPId toolbox is planned to be extended with the option for parallel simulation, where the optimization tasks are distributed to parallel workers in MATLAB distributed across multiple cores. It is expected that this would further increase simulation speed and produce results for parameter estimation problems faster. Additional examples will be added to the toolbox in the future using hardware-in-the-loop to compare the models to measurements from an automatic voltage regulator.

ACKNOWLEDGEMENTS

The authors thankfully acknowledge the support of Zdravko Rabuzin of Advensys Engineering Ltd. for providing the data of the hydroelectric power plant.

REFERENCES

ALSETLab (2020). RaPId: Rapid Parameter Identification. `https://github.com/ALSETLab/RaPId`.

Baudette, M., Castro, M., Rabuzin, T., Lavenius, J., Bogodorova, T., and Vanfretti, L. (2018). OpenIPSL: Open-Instance Power System Library — Update 1.5 to "iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations". *SoftwareX*, 7, 34–36.

Digilent (2020). Analog Discovery 2. `https://tinyurl.com/DigilentAD2`.

Fritzson, P. (2004). *Principles of Object Oriented Modeling and Simulation with Modelica 2.1*. doi:10.1109/9780470545669.

Garnier, H. and Gilson, M. (2018). Contsid: a matlab toolbox for standard and advanced identification of black-box continuous-time models. *IFAC-PapersOnLine*, 51(15), 688 – 693. doi:10.1016/j.ifacol.2018.09.203. 18th IFAC Symposium on System Identification SYSID 2018.

IEEE (2020). IEEE Recommended Practice for Excitation System Models for Power System Stability Studies. `https://tinyurl.com/IEEEStd421p5`.

Kalsi, K., Sun, Y., Huang, Z., Du, P., Diao, R., Anderson, K.K., Li, Y., and Lee, B. (2011). Calibrating multi-machine power system parameters with the extended Kalman filter. In *2011 IEEE Power and Energy Society General Meeting*, 1–8. doi:10.1109/PES.2011.6039224.

Kundur, P. (1994). *Power System Stability and Control*. McGraw-Hill.

MathWorks (2020a). Global Optimization Toolbox. `https://tinyurl.com/MatlabGOT`.

MathWorks (2020b). SysId Toolbox. `https://tinyurl.com/MSITX`.

Modelica Association (2020). Modelica Standard Library. `https://tinyurl.com/ModelicaMSL`.

Modelica Association Project (2020). FMI Standard. `https://fmi-standard.org/`.

Modelon (2018). FMI Toolbox User's Guide 2.6.4. Technical report, Modelon AB.

NASPI (2015). Model Validation Using Measurement Data. Technical report, North American Synchrophasor Initiative. `https://tinyurl.com/NASPImodelVal`.

Tamaševiciute, E., Tamaševicius, A., Mykolaitis, G., Bumeliene, S., and Lindberg, E. (2008). Analogue electrical circuit for simulation of the duffing–holmes equation. *Nonlinear Analysis*, 13(2), 241–252.

Taylor, C., Young, P., Tych, W., and Wilson, E. (2018). New developments in the CAPTAIN Toolbox for Matlab with case study examples. *IFAC-PapersOnLine*, 51(15), 694 – 699. doi:10.1016/j.ifacol.2018.09.202. 18th IFAC Symposium on System Identification SYSID 2018.

Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., and Martí, R. (2007). Scatter search and local nlp solvers: A multistart framework for global optimization. *INFORMS Journal on Computing*, 19(3), 328–340. doi:10.1287/ijoc.1060.0175.

Vanfretti, L., Baudette, M., Amazouz, A., Bogodorova, T., Rabuzin, T., Lavenius, J., and Goméz-López, F.J. (2016). RaPId: A modular and extensible toolbox for parameter estimation of Modelica and FMI compliant models. *SoftwareX*, 5, 144 – 149. doi:10.1016/j.softx.2016.07.004.