A Modelica Library for Continuous and Discrete Extremum Seeking for Static and Dynamic Systems

Joscha Müller Maxime Baudette Daniel Arnold Michael Sankur

Lawrence Berkeley National Laboratory
{joschamueller,baudette,dbarnold,msankur}@lbl.gov

Abstract

Extremum Seeking (ES) is an optimization scheme that has become a popular tool for addressing decision-making problems in settings where system models are unavailable or inaccurate and communications are unreliable. This paper presents an open source Modelica Extremum Seeking library that introduces different continuous and discrete ES controllers and examples for possible ES control applications. The controllers are available for Modelica and in the Functional Mock-up Interface (FMI) standard, which allows the models to be used in a variety of different software environments.

1 Introduction

Black-box optimization methods are an important class of algorithms used in situations where objective functions are difficult or expensive to evaluate, or are, perhaps, unknown to the optimizer. Many black-box optimization techniques rely on a local exploration of the action space to determine the best action to take. Within this subclass of algorithms, Extremum Seeking (ES) approaches are particularly useful as they require no knowledge of the system over which they are optimizing.

This “model-free" property of ES has made it attractive in a variety of applications, including homogeneous charge compression ignition (HCCI) engine optimization (Killingsworth et al., 2009), maximum power point tracking (MPPT), wind turbine optimization (Krstic et al., 2014), and control of autonomous robots (Zhang et al., 2007). Additionally, several books have been written on the topic (Zhang and Ordonez, 2012), (Ariyur and Krstic, 2003). Many of the authors of this work have utilized ES to manage power injections of distributed generation devices in the smart grid (Arnold et al., 2018).

In the ES scheme the optimizer (e.g., mobile robot, power generation source, parameter to be tuned) injects a small sinusoidal perturbation into the local action space. This perturbation, in turn, introduces an oscillation in the objective function value. With proper filtering to extract the oscillatory component of the objective function, the decision-maker can extract the gradient of the control input with respect to the objective. No explicit knowledge of the structure of the objective function is needed to obtain the gradient information, only a time series of measurements of the objective.

Our past research has explored several extensions of the family of ES algorithms, such as the simultaneous control of two setpoints using a single controller (Arnold et al., 2018), and the introduction of a decaying probe (Sankur and Arnold, 2019). More recently we started studying the impacts of communication delays and missing data in the system’s measurements on the performances of ES, following feedback from experimental implementation of the ES scheme in power system applications. This kind of studies requires a modeling environment that can easily implement more complex systems including both discrete and continuous dynamics. The Modelica language natively supports the modeling and simulation of complex cyber-physical systems and features many existing libraries containing models from many physical domains. It offers an attractive platform to further develop and study new variants of ES algorithms and their interaction with systems featuring more realistic characteristics (e.g., communication delays). This motivated the development of an ES library in Modelica. In this work, we present the library and examples of ES used to address several control/optimization problems. Models of different physical domains from the Modelica Standard Library were used to build a set of application examples, which are briefly highlighted in this paper.

The remainder of this paper is organized as follows. First, the working principle of ES control is introduced, followed by an overview of several examples of ES applied to domain-specific problems. Specifically, we discuss the optimization of a quadratic map, control of a spring-mass damper, speed control of a rolling wheel, and optimization of power injections in the smart grid. The paper concludes with a brief discussion of plans for future extensions to the ES Modelica library. The ES library was published as an open source project, and made available at `https://github.com/LBNL-ETA/ESL`.

Joscha Müller, Maxime Baudette, Daniel Arnold, and Michael Sankur are with the Grid Integration Group within the Energy Technologies Ares at the Lawrence Berkeley National Laboratory. This work was supported in part by the U.S. Department of Energy ARPA-E (DE-AR0000340) and Office of Energy Efficiency and Renewable Energy (DE-EE0008008 and DE-AC02-05CH11231).

(a) Continuous time implementation (`ES_BASIC`).

(b) Discrete time implementation (`discrete_ES_BASIC`).

Figure 2. Block diagrams for basic conventional ES Controllers.

the objective function and at least one ES logic block. contains several blocks for both the objective function and the ES logic. The objective function blocks allow the tracking of a target signal, or the regulation to keep the controlled variables within a defined interval. The different blocks also let the user include one or multiple signal(s) in the objective function. The ES logic blocks propose different implementations that are detailed in the upcoming sections. Finally, the library also features examples that are presented in Section 4.

3.1 Conventional Extremum Seeking Controller

The conventional Extremum Seeking controller (ES) is the most straightforward implementation of ES, as described in (Ariyur and Krstic, 2003). We built the block using the Modelica Standard Library, assembling the components as in similar fashion to Figure 1.

We have created several versions of this controller, implementing both continuous time and discrete time versions (see Figure 3). `ES_BASIC` is the basic version of ES in continuous time, shown in Figure 2a. `discrete_ES_BASIC` is a basic version of ES in discrete time, shown in Figure 2b. Both versions replace the bandpass filter with a high-pass filter, omit the gradient averaging operator, and omit setpoint limits.

`ES_ADV` is an advanced version of ES in continuous time, shown in Figure 3a. `discrete_ES_ADV` is an advanced version of ES in discrete time, shown in Figure 3b. Both `ES_ADV` and `discrete_ES_ADV` have a bandpass filter before demodulation, and a setpoint limiter. The user can choose to average the gradient in `ES_ADV`.

`ES_ADV_2D` is a 2-dimensional ES controller, in which two `ES_ADV` probe on the same frequency, with sinusoid phase offset by 2 (Arnold et al., 2018; Sankur and Arnold, 2019).

The reader is invited to read (Choi et al., 2002) for an analysis of a discrete time implementation.

3.2 Single Fixed-Step Extremum Seeking Controller

`discrete_ES_SFS` is the Single Fixed-Step Extremum Seeking Controller (SFSES), and is shown in Figure 3c. We based our implementation on the Modelica Standard Library and developed a new square wave for the probe and demodulation signals.

The SFSES updates its setpoint by a fixed step size, with the direction determined by a sign function inside the `convergence` block. The `convergence` block contains thresholds, set by the user, for switching, such that if the magnitude of the input signal is too small, the sign function will return 0. The user can implement a minimum setpoint limit and/or a maximum setpoint limit.

3.3 Multiple Fixed-Step Extremum Seeking Controller

`discrete_ES_MFS` is the Multiple Fixed-Step Extremum Seeking controller (MFSES), and is shown in Figure 3d. We based our implementation on the Modelica Standard Library and developed a new square wave for the probe and demodulation signals.

The MFSES is similar to the SFSES, but its setpoint update is an integer multiple of the step size. The `round` block in the MFSES rounds its input, which is the gradient estimate multiplied by a gain, to the nearest integer multiple of the step size. The `round` block may round its input to zero, such that the setpoint is held constant. The user can implement a minimum setpoint limit and/or a maximum setpoint limit.

3.4 Functional Mock-up Units

Modelica also supports the Functional Mock-up Interface (FMI) standard. The FMI standard is an independent standard for model-exchange (ME) and co-simulation (CS). It establishes a standard binary format, the Functional Mock-up Unit (FMU), that can be imported into other FMI compliant software tools. It allows to combine models from different languages / tools in a single simulation (Blochwitz et al., 2012).

We exported the controller models as FMUs, to extend the outreach of the library to users outside of the Modelica community. Each FMU was exported for both CS and ME simulation modes that are defined in the FMI standard.

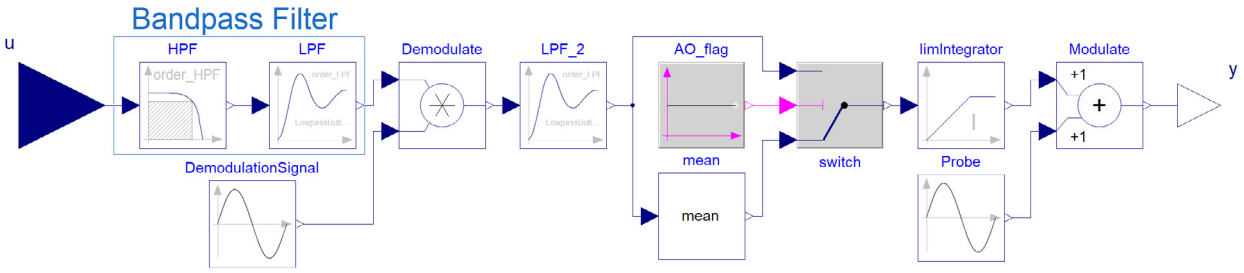
4 Examples

In this section, we present several examples in which one or more ES Controllers are employed to optimize an objective function, or implement control of an mechanical or electrical system.

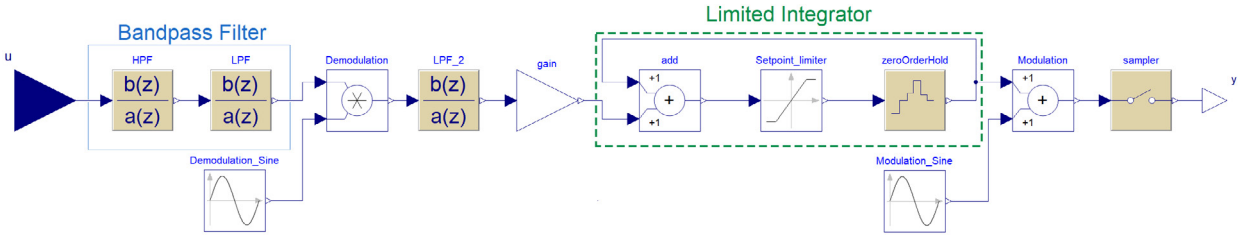
4.1 Quadratic Objective Function minimized by a Single ES Controller

In our first example, a single ES controller optimizes its setpoint to minimize the following quadratic function:

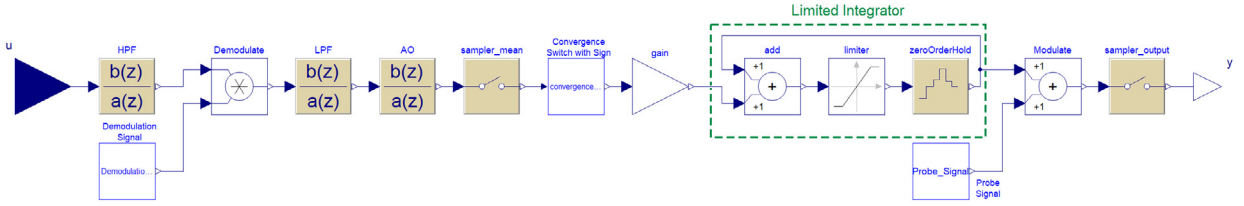
$$\Psi\ u \qquad u - 1\ ^{2}$$



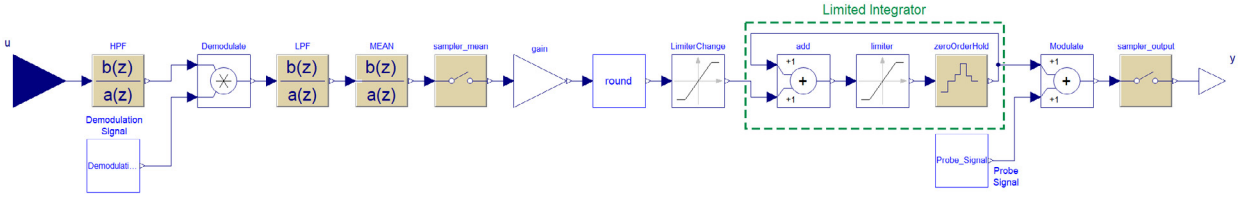
(a) Continuous ES (`ES_ADV`).



(b) Discrete ES (`discrete_ES_ADV`).



(c) Discrete Single Fixed-Step ES (`discrete_ES_SFS`).



(d) Discrete Multiple Fixed-Step ES (`discrete_ES_MFS`).

Figure 3. Block diagrams of several ES controllers of the library.

where u is the output of the ES controller. We ran three scenarios: one where a conventional ES minimizes the quadratic function, one where SFSES minimizes the quadratic function, and one where a MFSES minimizes the quadratic function, as seen in Figure 4. The controllers do not have knowledge of the system or objective function, but receive measurements of the objective function value.

Figure 5 shows simulation results of the three scenarios. All three ES controllers optimize their setpoint to minimize the objective function. SFSES takes the longest time to converge, as it can only update its setpoint by one probe amplitude at a user defined rate, in this case 2 probe cycles. Its oscillatory behavior after reaching the optimal value is due to the setpoint update switch algorithm, and can be eliminated by implementing a switching threshold on the averaged gradient estimate.

4.2 Single Quadratic Objective Function Minimized by Three ES Controllers in Parallel

In this example, shown in Figure 6, one conventional ES controller, one SFSES controller, and one MFSES controller operate in parallel to minimize the following function:

$$\Psi(u,v,w) = (u-1)^2 + (v-2)^2 + (w-3)^3,$$

where u is the ES output, v is the SFSES output, and w is the MFSES output. The controllers do not have knowledge of the objective function, but receive measurements of its value. Simulation results are given in Figure 7. This example shows that multiple ES controllers, and multiple types of ES controllers, can detect their own impact on an objective and optimize themselves, as long as the frequency of each controller is not equal to, or a multiple of, the frequency of any other controller.

 10.3384/ECP2016936 DOI

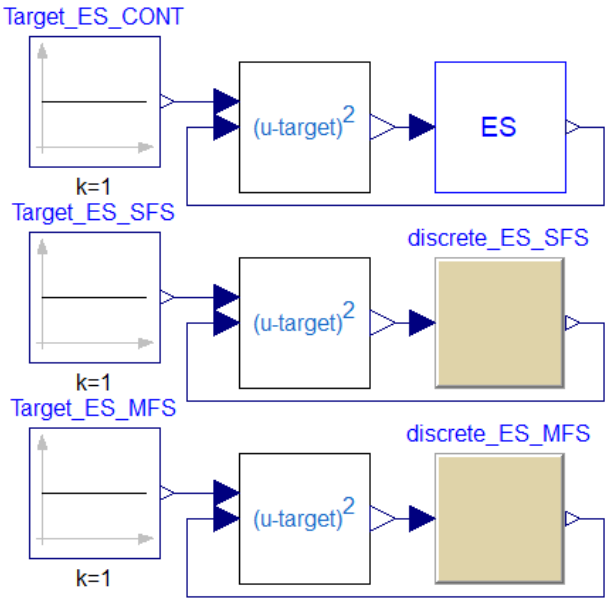


Figure 4. Block diagram for ES, SFSES, and MFSES controllers minimizing identical quadratic functions.

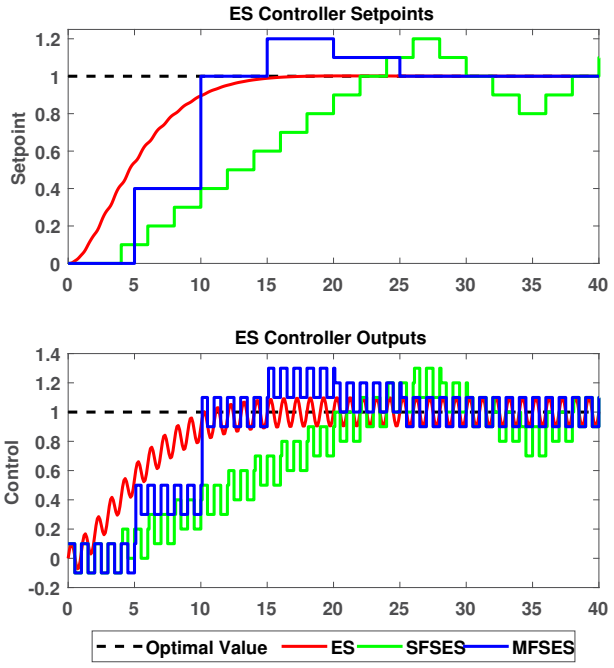


Figure 5. ES controller setpoints, and ES controller outputs for three separate simulations in which a single ES controller minimizes a simple quadratic function.

4.3 ES Control of Mass-Spring-Damper System

This example is an extension of the mass-spring-damper example from `Modelica.Mechanics.Translational.Examples.Damper` from the Modelica Standard Library. We extended the model with a position sensor, and a force applied to the mass and regulated by conventional ES, as in Figure 8. The objective function is:

$$\Psi = (y(u) - 5)^2,$$

where $y(u)$ is the position of the mass, and is a function of the ES regulated force u. Figure 9 shows that the controller converges to the optimal force value for the mass to be nearest to the desired position after roughly 100 seconds. The long rise time is due to the "slow" probe frequency, chosen to be a factor of 10 times slower than the natural frequency of the mass-spring-damper system, so as to avoid exciting the system at the natural frequency. ES convergence speed is dependent on several factors, one of which being probe frequency.

4.4 ES Control of Rolling Wheel Speed

The following example is an extension of the Modelica Standard Library model `Modelica.Mechanics.Rotational.Examples.RollingWheel`. The example features a wheel with an input torque, and a nonlinear friction torque, applied to it, as in Figure 10.

We replaced the `torqueStep` block with a controllable torque, regulated by ES. The translational speed of the wheel is measured by a speed sensor. The controller operates to minimize difference between the wheel's rotation speed and a target speed by controlling the torque as in:

$$\Psi(u) = (y(u) - 1)^2,$$

where $y(u)$ is the wheel speed, which is a function of the torque applied to the wheel u, regulated by ES.

To compare the performance of the ES, SFSES, and MFSES controllers, we performed three separate simulations in which each type of ES controller optimized the torque applied to the wheel. Figure 11 shows the results of this experiment. All three types of controllers are able to regulate the torque to achieve the desired wheel speed. The conventional ES controller converged to the optimal torque as its setpoint can be updated by any size. The SFSES controller exhibited oscillatory behavior around the optimal torque value, as its setpoint update consists of a fixed step in either direction, depending on the sign of the gradient value. The MFSES controller overshot the optimal torque value, and eventually converged to a steady value near the optimal. This is due to the gradient value threshold for setpoint updates.

4.5 Power System: Distribution Feeder

In the last example, the ES library was used to build a power system example similar to our previous works (Sankur and Arnold, 2019), where the aim was to track a reference for the active (P) and reactive (Q) power injections at the feeder head. We used models from the OpenIPSL project (Baudette et al., 2018), in particular a three-phase implementation of the IEEE 13 test model.

The example was build to show the possibility to combine several ES resources into a single control scheme.

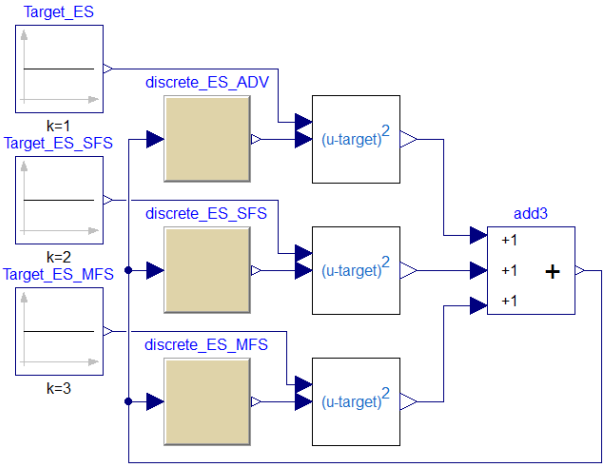


Figure 6. Block diagram of ES, SFSES, and a MFSES controllers operating in parallel to minimize a common quadratic objective function.

The modified model including the ES control scheme is shown in Figure 12. The scheme was designed arbitrarily to include ES resources throughout the feeder, connected to one, two, or three phases. In total, we added two three-phase resources (at nodes 632 and 680), one two-phase resource (at node 646), and one single-phase resource (at node 611) that were configured to be ES managed resources. These were connected to nine instances of the continuous 2D-ES (ESADV_2D) to manage both P and Q for each phase / resource Each individual ES was configured with the same gain corresponding to 0.2 kVA and a probing amplitude of 10 kVA. The probing frequencies were selected as $\sqrt{2}$, $\sqrt{3}$, $\sqrt{5}/2$, $\sqrt{7}/2$, $\sqrt{11}/3$, $\sqrt{13}/3$, $\sqrt{17}/4$, $\sqrt{19}/4$, $\sqrt{23}/4$.

The objective considered in this experiment was to reach different P and Q targets for each phase at the feeder head. We prepared a block to compute and extract the P and Q measurements of each phase, and placed it at the feeder head (node 650). The objective block was added and connected to constant targets that were set for P and Q for each phase respectively. The target values for P_a, P_b, P_c, Q_a, Q_b, and Q_c are chosen arbitrarily to 1, 1.5, 2 MW and -0.2, -0.3, -0.1 MVAr respectively for the purpose of the example, yielding the following objective function:

$$\Psi = (P_a - 1)^2 + (P_b - 1.5)^2 + (P_c - 2)^2 + (Q_a + 0.2)^2 + (Q_b + 0.3)^2 + (Q_c + 0.1)^2$$

The common objective was broadcast to every ES resources.

The example was initialized with all ES resources at zero The simulation was executed to reach time instant $t = 60s$ to let all the variables reach their target value. Figure 13 shows that the ES control sccheme is working as expected and all target values are reached. Note that the computation time of this particular example can be sharply reduced by toggling the *DAEsolver* mode in Dymola, which speeds up larger differential and algebraic equation systems, such as that of power system models

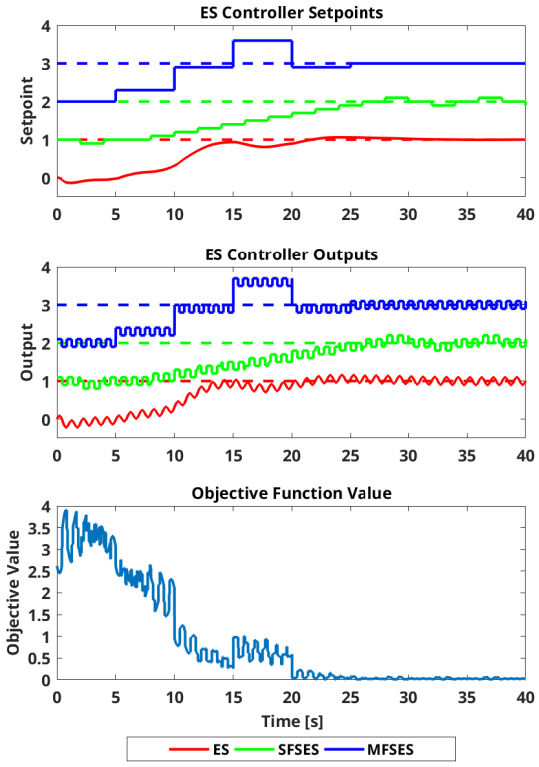


Figure 7. Controller setpoints, controller outputs, and objective function value for the scenario in which ES, SFSES, and MFSES controllers operate in parallel to minimize a quadratic objective function. Dashed lines represent the optimal value for the corresponding controller.

5 Conclusion

In this paper, we introduce an open-source Modelica package that implements several variants of Extremum Seeking (ES). ES offers an attractive solution for model-free optimization and control, and is applicable to a wide range of problems. In this paper we introduced an open-source package that implements several variants of ES stemming from the Author's previous work in grid applications. The implementation was carried out in Modelica, providing a flexible framework to model multiple types cyber-physical systems. The Modelica ES implementation allowed us to prepare examples for different domains that we included in the library. Thanks to Modelica's ability to combine continuous and discrete models, it was also possible to compare different ES variants in practical applications and its impact on the physical systems being controlled.

Our efforts focused on using a generic design to broaden the target audience to different domains. The ES control scheme was broken down into blocks that sepa-

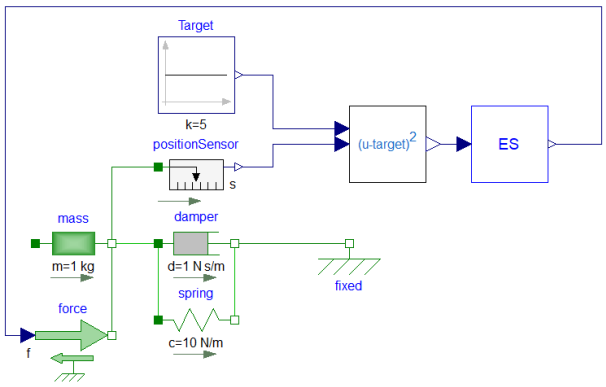


Figure 8. Block diagram of the Mass-Spring-Damper example.

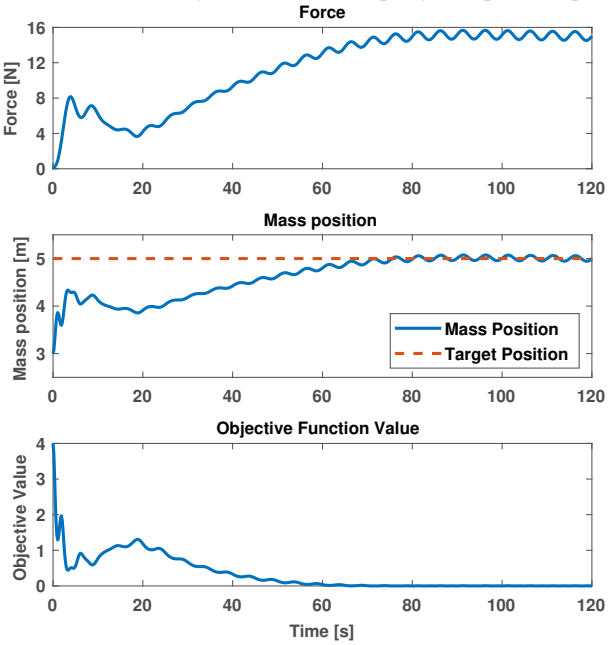


Figure 9. Force applied to the mass, mass position, and objective function value for the mass-spring-damper example.

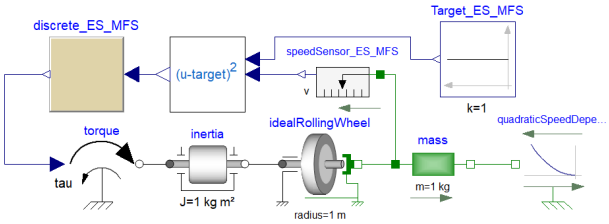


Figure 10. Block diagram of the rolling-wheel example.

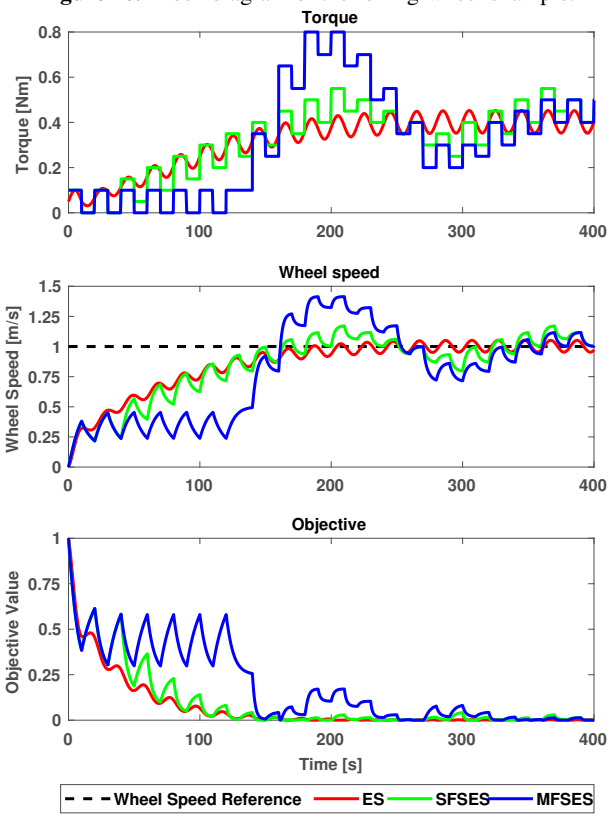


Figure 11. Torque, translational wheel speed, and objective function for the rolling wheel example, for three scenarios with ES, SFSES, or MFSES.

rate the objective function from the ES logic. This allows a flexible assembly of the inter-compatible blocks to suit the needs of different research communities of various domains. Finally, we included compiled versions of the ES blocks in the form of FMUs to further extend the possible applications of the library to any FMI compliant simulation tools.

We plan to extend and further develop our Modelica ES library in several key ways. We plan to add equilibrium based switching for probe amplitude decay, as in the work of several of this work's authors (Sankur and Arnold, 2019). A second key area is to add an estimator for the phase difference between controller output (system input) and objective function measurements, This is especially important when using ES to optimize or control dynamic systems. It also has applications for control systems that rely on digital network communications that introduce non deterministic delays. We also plan to implement more examples in different domains, including optimization of heat and fluid models.

References

Kartik B Ariyur and Miroslav Krstic. *Real-time optimization by extremum-seeking control*. John Wiley & Sons, 2003.

D. B. Arnold, M. D. Sankur, M. Negrete-Pincetic, and D. Callaway. Model-free optimal coordination of distributed energy resources for provisioning transmission-level services. *IEEE Trans. Power Syst.*, 33(1):817–829, Jan. 2018. ISSN 0885-8950. doi:10.1109/TPWRS.2017.2707405.

Maxime Baudette, Marcelo Castro, Tin Rabuzin, Jan Lavenius, Tetiana Bogodorova, and Luigi Vanfretti. OpenIPSL: Open-Instance Power System Library–Update 1.5 to "iTesla Power

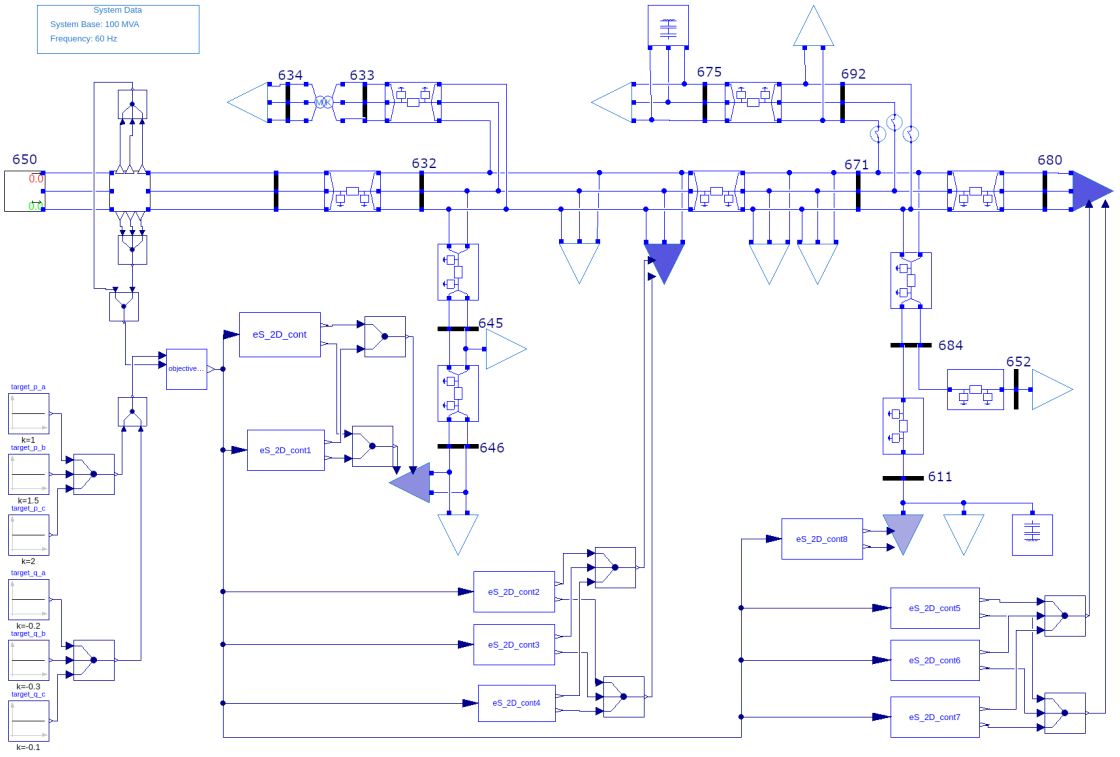


Figure 12. Block diagram of the IEEE 13 Example (ES resources highlighted in blue).

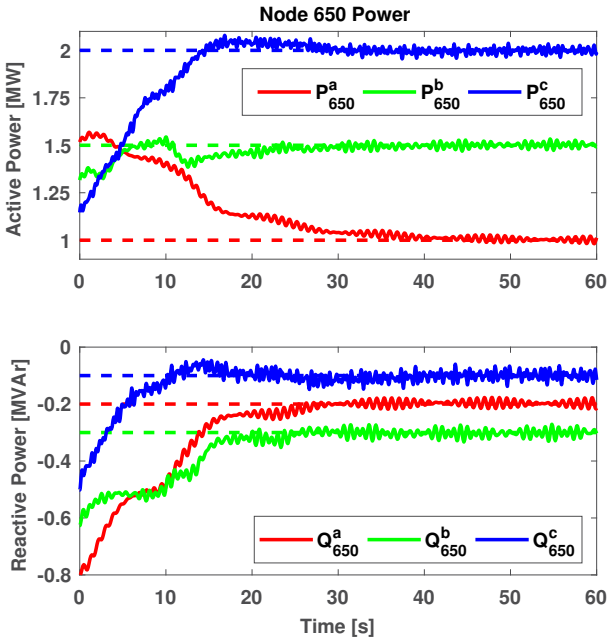


Figure 13. Active and Reactive power for each phase at Node 650 of the IEEE 13 node test feeder. Dashed lines represent the corresponding target value.

Systems Library (iPSL): A Modelica library for phasor time-domain simulations". *SoftwareX*, 7:34–36, 2018.

Torsten Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, et al. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 076, pages 173–184. Linköping University Electronic Press, 2012.

Joon-Young Choi, Miroslav Krstic, Kartik B Ariyur, and Jin Soo Lee. Extremum seeking control for discrete-time systems. *IEEE Transactions on automatic control*, 47(2):318–323, 2002.

N. J. Killingsworth, S. M. Aceves, D. L. Flowers, F. Espinosa-Loza, and M. Krstic. HCCI engine combustion-timing control: Optimizing gains and fuel consumption via extremum seeking. *IEEE Transactions on Control Systems Technology*, 17(6):1350–1361, Nov 2009. doi:10.1109/TCST.2008.2008097.

M. Krstic, A. Ghaffari, and S. Seshagiri. Extremum seeking for wind and solar energy applications. In *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pages 6184–6193, June 2014. doi:10.1109/WCICA.2014.7053780.

Michael Sankur and Daniel Arnold. Extremum seeking control of distributed energy resources with decaying dither and equilibrium-based switching. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

Chunlei Zhang and Raul Ordonez. *Extremum-Seeking Control and Applications A Numerical Optimization-Based Approach*. 01 2012. ISBN 978-1-4471-2223-4. doi:10.1007/978-1-4471-2224-1.

Chunlei Zhang, Daniel Arnold, Nima Ghods, Antranik Siranosian, and Miroslav Krstic. Source seeking with nonholonomic unicycle without position measurement and with tuning of forward velocity. *Systems & control letters*, 56(3): 245–252, 2007.