INL/EXT-21-63204
Revision 0

Status Report on the INL IES Plug-and-Play Framework

June | 2021

Andrea Alfonsi
Konor L Frick
Cristian Rabiti
Shannon Bragg-Sitton
Idaho National Laboratory

INL/EXT-21-63204
Revision 0

Status Report on the INL IES Plug-and-Play Framework

Andrea Alfonsi
Konor L Frick
Cristian Rabiti
Shannon Bragg-Sitton
Idaho National Laboratory

June 2021

Idaho National Laboratory
Integrated Energy Systems
Idaho Falls, Idaho 83415

http://www.ies.inl.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Page intentionally left blank

ABSTRACT

This report discusses the advancements and status of the flexible plug-and-play framework development currently ongoing that aims to integrate Modelica/Dymola with the Risk Analysis and Virtual ENvironment (RAVEN) software in terms of both Functional Mock-Up Interface (FMI)/Functional Mock-Up Unit (FMU) construction and usage, which aims to ease the sharing and simulation of complex dynamic models.

This report discusses the FMI/FMU development advancements, overall focusing on the deployment of methodologies for RAVEN to export and use FMI/FMU of both Python-based models and advanced AI-constructed algorithms.

Page intentionally left blank

CONTENTS

FIGURES

Page intentionally left blank

ACRONYMS

AI	Artificial Intelligence
API	Application Program Interface
CPU	Central Processing Unit
FMI	Functional Mock-Up Interface
FMU	Functional Mock-Up Unit
FORCE	Framework for Optimization of ResourCes and Economics ecosystem
IES	Integrated Energy Systems
INL	Idaho National Laboratory
NHES	Nuclear Hybrid Energy Systems
RAVEN	Risk Analysis and Virtual Environment
ROM	Reduced-order model
SQA	Software Quality Assurance

Page intentionally left blank

1. INTRODUCTION

The Integrated Energy System (IES) Program is continuing its effort to design, improve, and enhance the modeling and simulation capabilities aimed at assessing the economic viability of integrated energy systems. Among the different activities, a key component of the recent research efforts is about the creation of an infrastructure for the flexible interaction of Modelica/Dymola models [1] with the Risk Analysis and Virtual ENvironment (RAVEN) ecosystem [2],[3].

In the past year, two additional status [4],[5] and a detailed development [6] reports have been delivered with the description of the effort to create a "plug-and-play" repository of process models using Modelica, Functional Mock-Up Interfaces (FMIs), and Functional Mock-Up Units (FMUs).

Since most of the effort aimed to adapt and facilitate the export of FMI/FMU from Modelica/Dymola models has been presented in the recent March release [6], the goal of this document is to report the advancements and future for the deployment of AI-based FMI/FMU aimed of achieving of multi-fidelity plug and play framework.

2. RAVEN FMI/FMU DRIVING SYSTEM DEVELOPMENT

In previous milestone reports [4],[5] and [6], the successful execution of the FMIs and FMUs using external Python-based framework (FMpy [7] and PyFMI [8]) has been demonstrated alongside an initial development within the RAVEN framework.

As reported in [6] the coupling with system and physical models is performed by a model entity application program interface (API) inside RAVEN. The model entity represents a "transfer function" between the input and the output space. The RAVEN framework provides APIs for the following main model categories:

- Codes
- Externals
- Reduced order models
- Hybrid models
- Ensemble models
- Postprocessors

The external model (see Figure 1) allows the developer (for plugins) or user to create, in a Python module or platform, a direct coupling with, a model coded in Python (e.g., a set of equations representing a physical model, connection to another code, control logic). Once the external model is constructed, it is interpreted and used by RAVEN and, at run-time.

As mentioned, the development of the FMI/FMU driving system is based on the *ExternalModel* entity in RAVEN. As briefly reported in the previous section, the external model (see Figure 1) enables developers to create, in a *Python* module or platform, a direct coupling with a model coded in *Python* (e.g., a set of equations representing a physical model, connection to another code, and control logic). Once the external model is constructed, it is interpreted and used by RAVEN, ultimately becoming, at run-time, part of RAVEN itself.

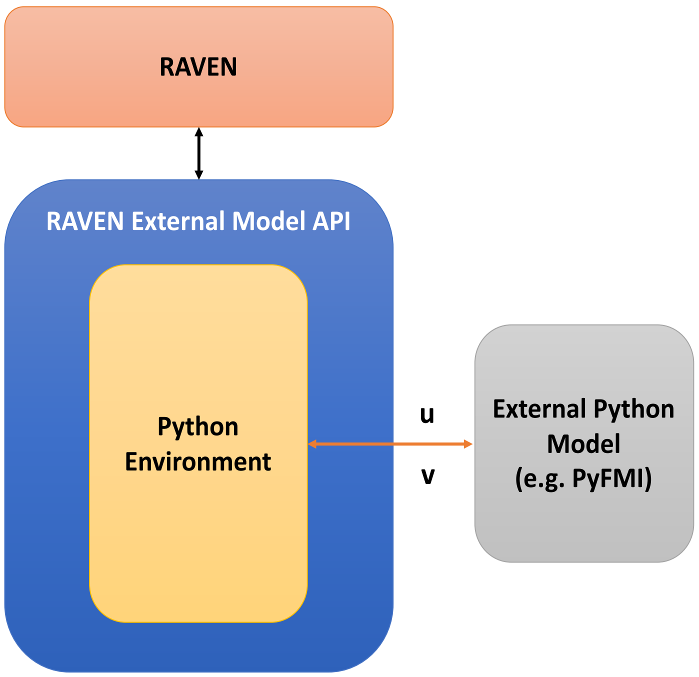
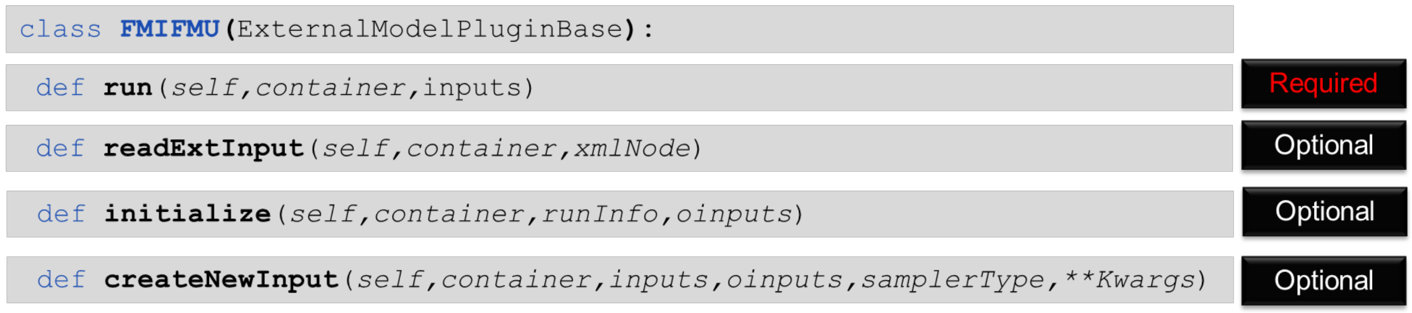


Figure 1. External model API.



```
class FMIFMU(ExternalModelPluginBase):
  def run(self,container,inputs)
  def readExtInput(self,container,xmlNode)
  def initialize(self,container,runInfo,oinputs)
  def createNewInput(self,container,inputs,oinputs,samplerType,**Kwargs)
```

Figure 2. FMI/FMU model skeleton in RAVEN.

The *ExternalModel* API (*ExternalModel* plugin) is used as a platform, in RAVEN, to deploy a native driver for models using the FMI/FMU protocol. Figure 2 shows a snapshot of the “wrapper” that was developed and is being deployed in RAVEN. The “*FMIFMU*” RAVEN model implements a generalized method—based on the RAVEN API and syntax—to import, execute, and process the results of any model compatible with the FMI/FMU standard (e.g., Dymola, Open-Modelica, Ansys, etc.). The model API deploys the following methods:

- ***run***: The run method (the only required method in the API) aims to execute the FMU (FMI) for a given input coordinate (or input perturbation). The ***run*** method represents the link between RAVEN and the FMI/FMU model. The method both executes and collects the results that will be then stored in the object “container,” ready for processing by RAVEN. The run method, based on a flag sent in the “container” object, can either run the full simulation (e.g. from ***startTime*** seconds to ***stopTime***) or a single time-step (***stepSize***); in the second case, the model can communicate with other models at the time-step level.

- ***readExtInput***: This method oversees reading the user-define input for the FMI/FMU that needs to be driven. It collects the following information (expandable in the future, if needed):
    - ***startTime***: The start time of the driven FMU (e.g., 0.0 seconds)
    - ***stopTime***: The stop time of the driven FMU (e.g., 60 seconds)
    - ***stepSize***: The time step size to use for the calculation (e.g., 1.e-2 seconds)
    - ***inputVariables***: A list of the input variables (e.g., demand)
    - ***outputVariables***: A list of the output variables (e.g., power level)
    - ***fmuFile***: The FMU location (e.g., */path/to/myFmu.fmu*)
- ***initialize***: This method is invoked right before the model is executed. This method aims to load the FMI/FMU, instantiate the class, and initialize its settings.

  This method is also in charge of performing error checking of the user-defined settings/options. In here, the sanity check of the FMU is performed.
- ***createNewInput***: This method, in case a sampling strategy is requested by the user, is responsible for translating" the RAVEN info (e.g., the values of sampled variables) into the FMI/FMU syntax.

```
<Models>
  <ExternalModel name="fmu" subType="FMIFMU">
    <variables> demand, time, SES_generator_P_flow</variables>
    <startTime> 0.0 </startTime>
    <stopTime> 14400</stopTime>
    <stepSize> 1.0 </stepSize>
    <inputVariables> demand </inputVariables>
    <outputVariables> SES_generator_P_flow</outputVariables>
    <fmuFile> GTTPfmu.fmu </fmuFile>
  </ExternalModel>
</Models>
```
Figure 3. External model FMIFMU example RAVEN input file.

Figure 3 shows an example of the portion (in XML) of the RAVEN input file required to use the *FMIFMU* wrapper. This XML block is the one processed by the method "*readExtInput*.". Independently on the type of FMI/FMU that the model will import and use, the input file specifications do not change; the FMIFMU wrapper will collect the information (co-simulation or model exchange) directly from the FMU (i.e., the **<fmuFile>**) after loading.

In summary, the "*FMIFMU*" RAVEN model oversees loading, initializing, and executing the FMI/FMU of interest and allows RAVEN to drive it as any other Model.

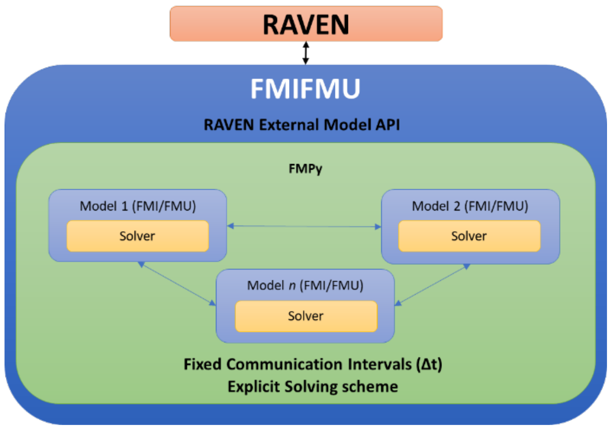


Figure 4. FMI/FMU co-simulation protocol coupled with RAVEN.

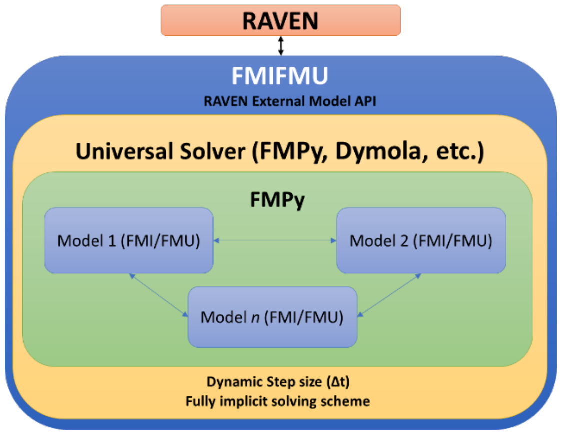


Figure 5. FMI/FMU model exchange protocol coupled with RAVEN.

2.1 RAVEN FMI/FMU coupling scheme of interest.

Depending on the type of protocol for the FMI or FMU of interest, two coupling schemes in the *FMIFMU* wrapper were developed. Both schemes are encapsulated in the same wrapper and are executable via the model API in RAVEN.

Figure 4 shows the coupling scheme for FMIs/FMUs when the co-simulation protocol must be used; RAVEN interacts with the different models via the *FMIFMU* wrapper that uses FMPy to import and interact with the FMUs. In this coupling scheme, RAVEN “perceives” the models imported via FMIs/FMUs just as it would any other external model or code. This protocol is indicated when the models to connect are loosely coupled (multi-physics feedbacks are not strong and/or the physics dynamic of the different models act on different time scales, e.g., seconds vs. hours or days).

On the other end, Figure 5 shows the coupling scheme for FMIs/FMUs when the model exchange protocol is used; in this configuration, RAVEN can *potentially* interact with the universal solver that aims to solve all the models (compatible with the FMI/FMU protocol, in this case Dymola). This coupling scheme is preferrable when the models are highly nonlinear, and the models are tightly coupled with fast moving dynamics. However, additional research is needed to make this style of coupling a reality.

The coupling schemes are crucial for the development of the Master Simulator platform that is currently under development (And will be briefly discussed in section 8).

3. DEPLOYMENT OF RAVEN FMI/FMU EXPORTER FOR AI-BASED ANALYSIS ACCELLERATIONS

RAVEN provides APIs for different model categories, among which are the ROM, AI-based algorithms. To deploy the acceleration of IES analysis, the ROM (AI) entity is key. Indeed, the ROM is aimed at higher fidelity surrogate and system simulator-based models (for specific and limited operational domain) with a set of faster-execution equations that allow for the prediction of Figures of Merit (of interest) in a span of milliseconds.

3.1 RAVEN AI construction

Figure 6 illustrates the standard process of constructing (via optimization) RAVEN surrogate (AI) models. The surrogate model of interest is trained on a dataset, and its hyper-parameters (i.e., parameters and characteristics of the surrogate model of interest) are tuned to maximize the accuracy in predicting the figure(s) of merit (FOMs) of interest. The accuracy is assessed by applying statistical methodologies (i.e., cross-validation), which consists of randomly portioning the dataset into “training” and “testing” datasets. The “training” dataset is used for constructing the surrogate model, and its prediction is compared with the “testing” dataset. The prediction accuracy is then assessed using distance metrics (e.g., R2 score) between the surrogate model and the testing dataset.

The so-constructed surrogate models allow for fast evaluation of the dynamics (or steady state) of the FOMs of interest. Therefore, such models can accelerate analyses (greatly reduce the computational time), by replacing high-fidelity physical models with a ROM representation.

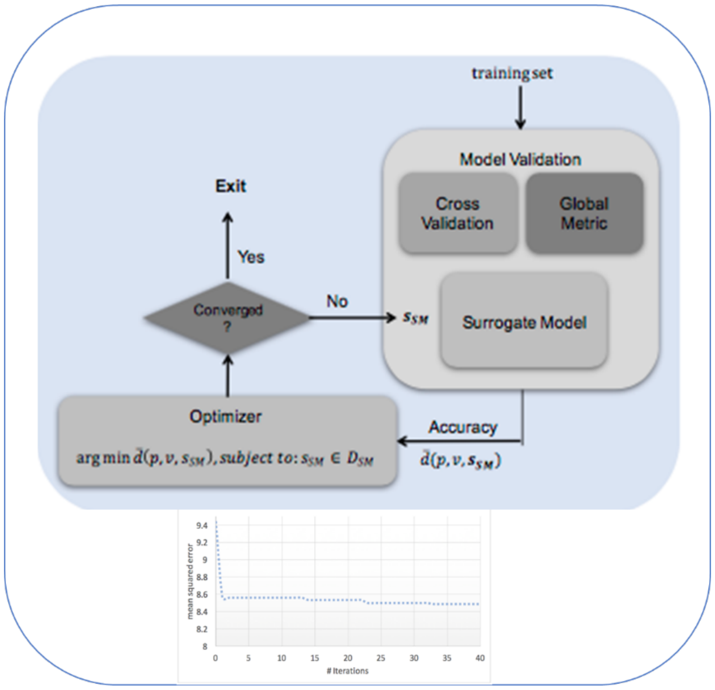


Figure 6. Construction process for surrogate models in RAVEN.

3.2 Development of FMI/FMU exporting capabilities for RAVEN AI

To exploit RAVEN AI capabilities, a workflow to export trained (constructed) ROMs using the FMI/FMU protocol was developed in RAVEN.

The exporting of RAVEN AI is performed according to the following two steps:

1) Exploit the native RAVEN serialization system, which is responsible for serializing (i.e., saving in a binary file) already-trained surrogate models that can be loaded in external (*Python*-based) packages (outside RAVEN).
2) Use and extend the PythonFMU library (https://github.com/NTNU-IHB/PythonFMU), which is a lightweight framework that enables the packaging of *Python* 3 code as co-simulation FMUs (following FMI version 2.0).

To deploy any model in an FMI/FMU-compatible framework, that model (i.e., ROM) must be able to be inquired at each "time step," meaning that the model must allow for execution as an integrated model and not as a "black-box simulation". To achieve this goal, the RAVEN ROM APIs were upgraded by implementing a "method" to solve the surrogate model at each time step. This modification, in conjunction with the two steps reported above, allows for RAVEN ROM models to be exportable as FMI/FMUs.

Once the RAVEN AI is trained following the standard process reported in section 3.1, it can be finally exported following the steps reported in Figure 7. An example of the RAVEN input blocks is reported in Figure 8, where:

- In the ***<Models>*** node, the RAVEN ROM (AI) is shown.
- In the ***<Files>*** node, the output FMI/FMU filename is specified.
- In the ***< Steps>*** node, the trained ROM (input) is exported as FMI/FMU (output).

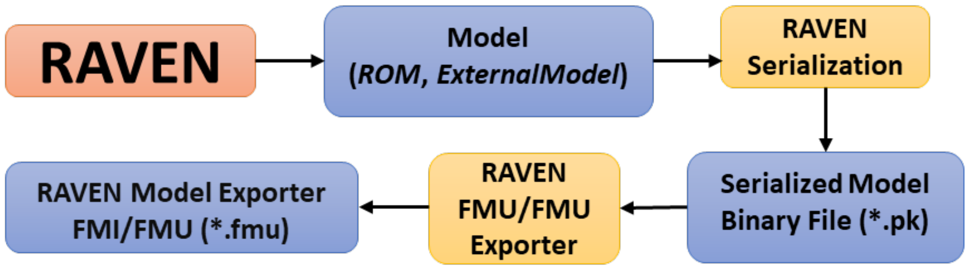

RAVEN
Model (*ROM, ExternalModel*)
RAVEN Serialization
Serialized Model Binary File (*.pk)
RAVEN FMU/FMU Exporter
RAVEN Model Exporter FMI/FMU (*.fmu)

Figure 7. RAVEN AI FMI/FMU exporting process.

```
<Models>
  <ROM name="GTTProm" subType="SciKitLearn">
    …
  </ROM>
</Models>
<Files>
  <Input name="FMU" type="fmu"> GTTProm.fmu </Input>
</Files>
```

```
<Steps>
  <IOStep name="romDump">
    <Input class="Models" type="ROM"> GTTProm </Input>
    <Output class="Files" type=""   > FMU     </Output>
  </IOStep>
</Steps>
```

Figure 8. Example RAVEN input file to export AI as FMIs/FMUs.

The FMI/FMU exporting capability allows for the coupling of RAVEN models, exported as FMI/FMU, in tandem with any Dymola/Modelica (in general) and HYBRID (in particular) physical models (see Figure 9).

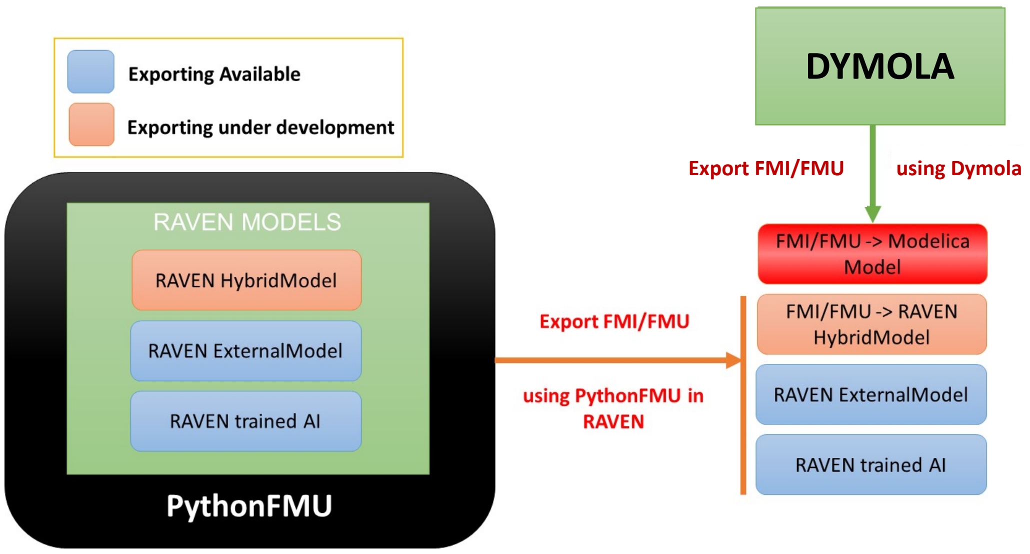


Figure 9. RAVEN's current FMI/FMU exporting capabilities.

4. Integrated Energy Park Demonstration Case

To demonstrate the capabilities described in this report, a test case on an integrated energy park was conducted. The integrated energy park, shown in Figure 10, consists of a nuclear reactor, electric batteries, and a natural gas turbine. The natural gas turbine is the component to be exported as an FMU. The natural gas peaking turbine is replaced with its own FMU from three different sources: the Dymola FMU in both model exchange and co-simulation, then a RAVEN-based surrogate using co-simulation mode.

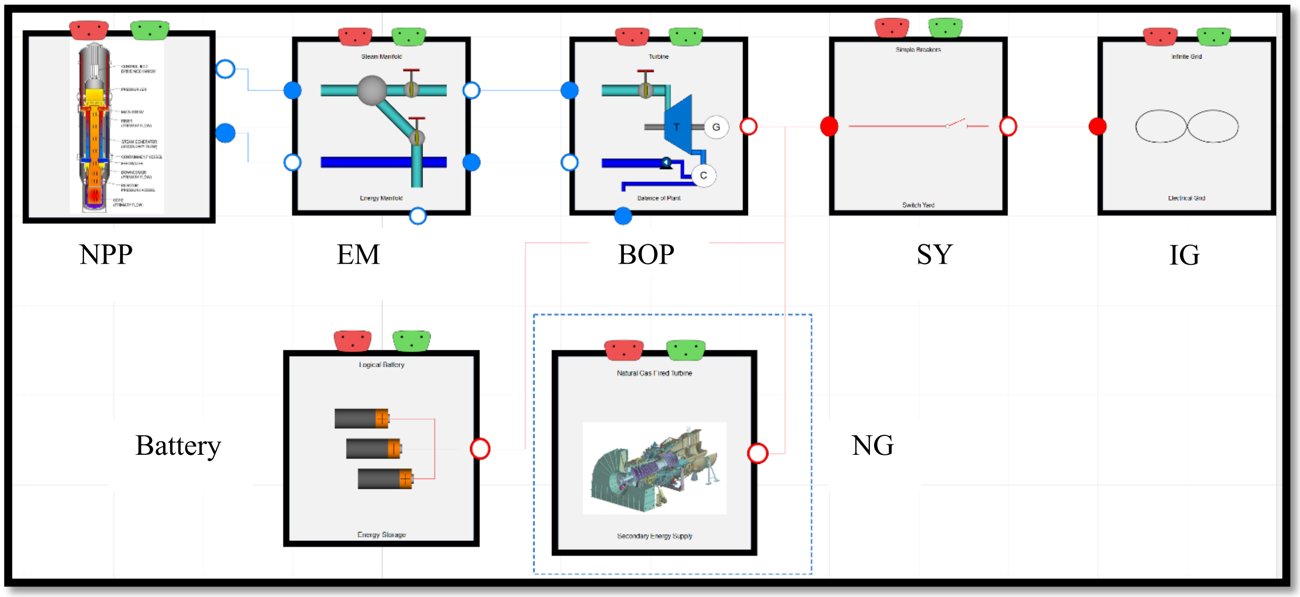


Figure 10. Integrated energy park consisting of a nuclear reactor (NPP), Energy Manifold (EM), Balance of Plant (BOP), Switch Yard (SY), Electric Batteries (Battery), Infinite Grid (IG), and a Natural Gas turbine (NG). The natural gas turbine is to be exported as an FMU.

4.1 FMI/FMU Creation and Use within Dymola

As detailed in [6], the procedure of exporting Dymola components as FMI/FMU requires modifications using adaptors to ensure that all the variables contained in the flow ports are realigned into real input/output variables. Once the model has been exported, as FMI/FMU, using the Dymola interface, it can then be re-imported into the program and can replace the natural gas turbine model (see Figure 11).

Using this version of the FMI/FMU, three separate 5-hour simulations were run: one with Modelica-only input, one with a co-simulation version of the gas turbine, and one in model exchange mode. The results of this simulation set are depicted in Figure 12. Over the course of the full 5-hour simulation, the results are all in near-perfect agreement with the setpoints, with model exchange and Dymola results being basically identical, and co-simulation being only as accurate as the communication step of 1 second would allow.

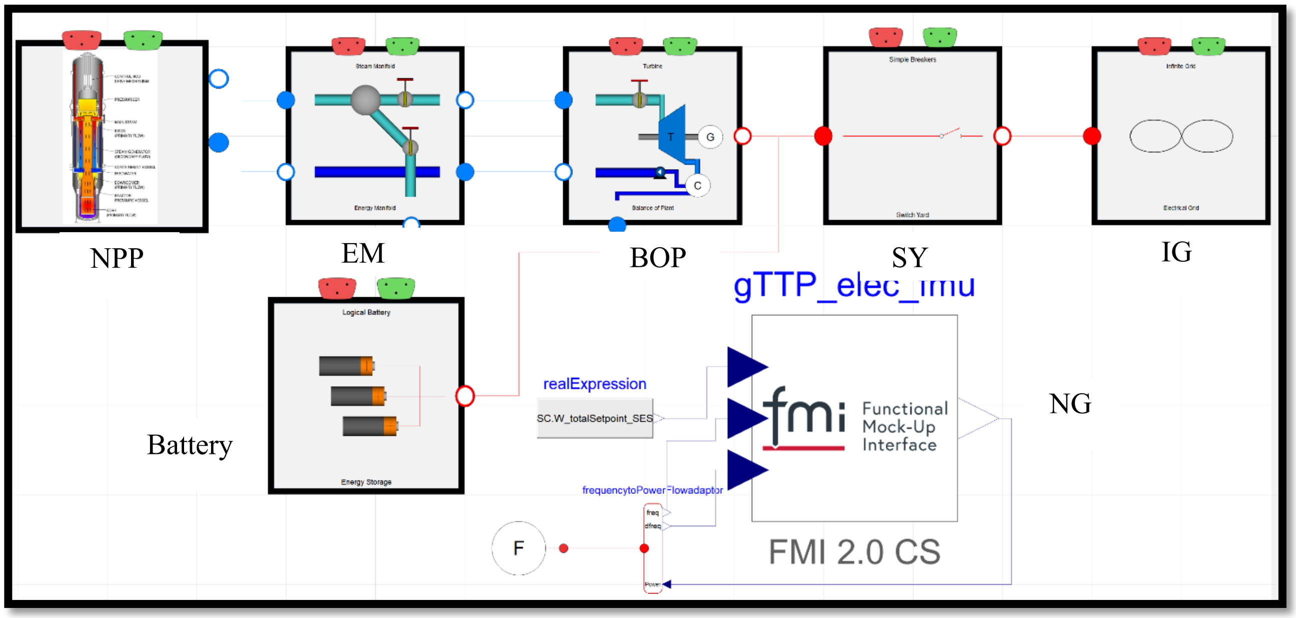
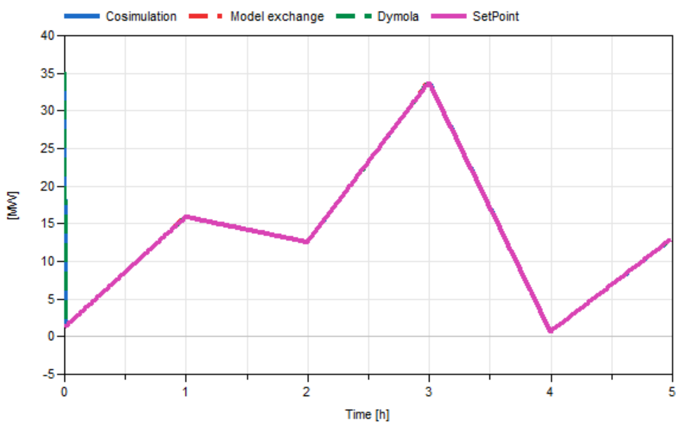


Figure 11. Integrated energy park consisting of a nuclear reactor (NPP), Energy Manifold (EM), Balance of Plant (BOP), Switch Yard (SY), Electric Batteries (Battery), Infinite Grid (IG), and a Natural Gas turbine (NG)..The natural gas turbine replaced by a co-simulation FMU.



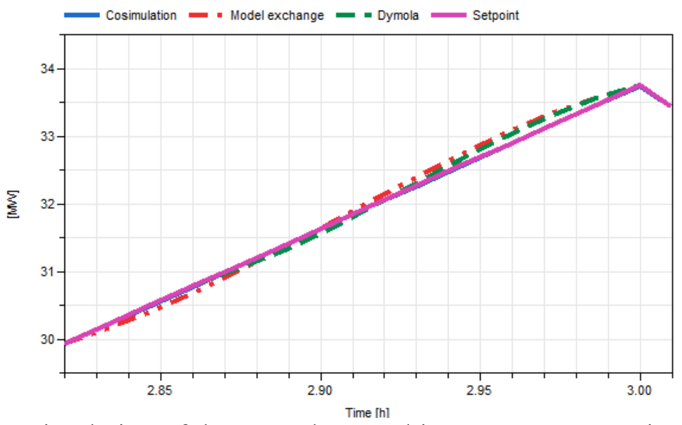


Figure 12. Top) Five-hour simulation of the natural gas turbine power vs. setpoint demand for the integrated energy park in regard to Modelica-only model, co-simulation FMI, and model exchange FMU.

Bottom) Closeup shot of the turbine demand vs. turbine output for the different FMI versions. Note that all agree reasonably well. Co-simulation communication interval = 1 second.

4.2 Creation of Surrogate Using RAVEN

Due to the large increase in simulation time, the relative issues with co-simulation initialization routines, and the fact that there is no feedback to the rest of the grid, the FMI created for use in the RAVEN surrogate training was reduced to having only a single input (turbine demand) with no connected outputs. This setup allows the natural gas turbine to keep all the initialization pieces of the "infinite" grid self-contained, thus drastically improving the initialization routine and system robustness. Since the turbine power is a variable given by the FMU, and no feedback is used in other units' control systems, the turbine power was not required to be an external variable for the initial export. The FMI/FMU (GTTP.fmu) exported to RAVEN is shown in Figure 13.

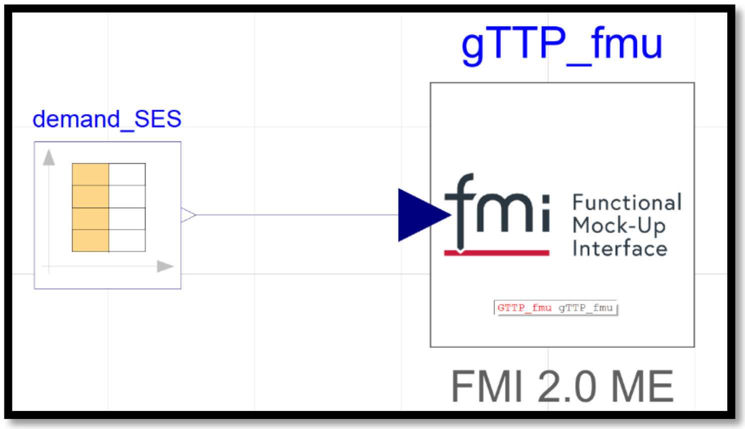


Figure 13. Simplified model of the FMI for RAVEN surrogation.

To construct a RAVEN-based AI to surrogate the response of the turbine component, the *FMIFMU* RAVEN importer described in previous sections was used to drive the Dymola-exported FMI/FMU model.

Since the turbine's response to changes in the demand is very quick (extremely limited inertia) and almost perfectly linear, a Support Vector Regressor with linear was selected for surrogating the response. The turbine FMI/FMU GTTP.fmu was loaded via the *FMIFMU* RAVEN importer and its demand sampled (1,000 Monte Carlo samples) between 0 and 35 MW to capture the model's full domain of variability. Finally, the Support Vector Regressor was trained (constructed) and exported to a "brand-new" FMI/FMU (GTTProm.fmu) by the RAVEN FMI/FMU exporter (co-simulation), as described in Section 3.2.

To validate the RAVEN AI FMI/FMU, a cross-validation assessment was performed in RAVEN, and, due to the pure linearity of both the turbine and AI models, its average R2 score was >0.99. This is further demonstrated in Figure 14 and Figure 15 which show a comparison of the Dymola-generated turbine FMI/FMU and the RAVEN AI FMI/FMU, with the models demonstrating good agreement.

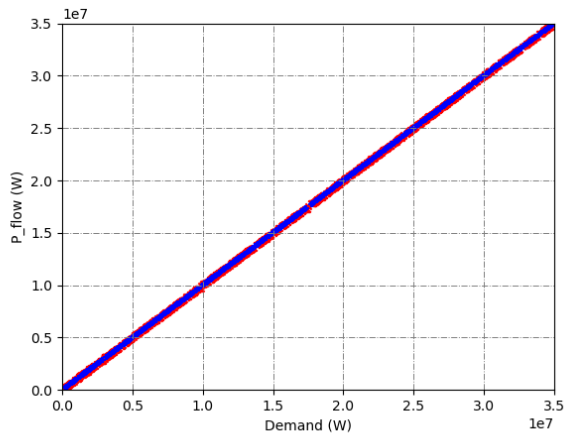


Figure 14. Comparison of the Dymola GTTP.fmu response (P_flow) and the RAVEN AI-based GTTProm.fmu.

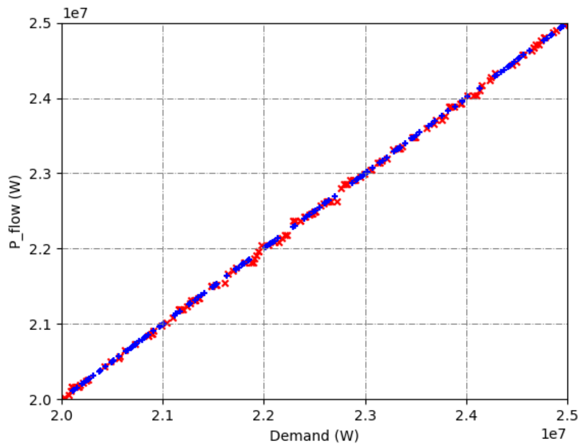


Figure 15. Closeup of the comparison of the Dymola GTTP.fmu response (P_flow) and the RAVEN AI-based GTTProm.fmu.

4.3 Comparison of Results

To demonstrate the concept of the “plug-and-play” framework, along with the usage of AI for accelerated analysis, the integrated energy park model was simulated, both using the original Dymola model (FMI/FMU) for the gas turbine and using the RAVEN AI-based model.

Figure 16 and Figure 17 show the setup of the integrated energy park along with the detailed Dymola FMI/FMU and the RAVEN AI-based FMI/FMU, respectively. Both models were simulated in an ad-hoc *Python* code (master simulator) using the FMPy package.

Using the above-mentioned FMI/FMU setup, the two 5-hour simulations were run in the master simulator (*Python* code using FMPy). Since the RAVEN AI-based FMI/FMU can be evaluated in mere milliseconds, the simulation of the setup with the AI was much faster (~20%) to complete, making the computation time for the turbine evaluation completely negligible; indeed, the AI FMI/FMU almost zeroed out the CPU time for the turbine simulation, and the totality of the CPU time was used to simulate the remaining systems in the integrated energy park, which were more complex and computationally intensive.

Figure 18 and Figure 19 show a comparison of the turbine responses in the integrated energy park using the Dymola FMI/FMU and the RAVEN AI-based FMI/FMU. Over the course of the full five-hour simulation, all the results were in near-perfect agreement with the setpoints. The results show that the setup using the RAVEN AI FMI/FMU outperformed (in terms of speed) the Dymola model, with no loss of accuracy.

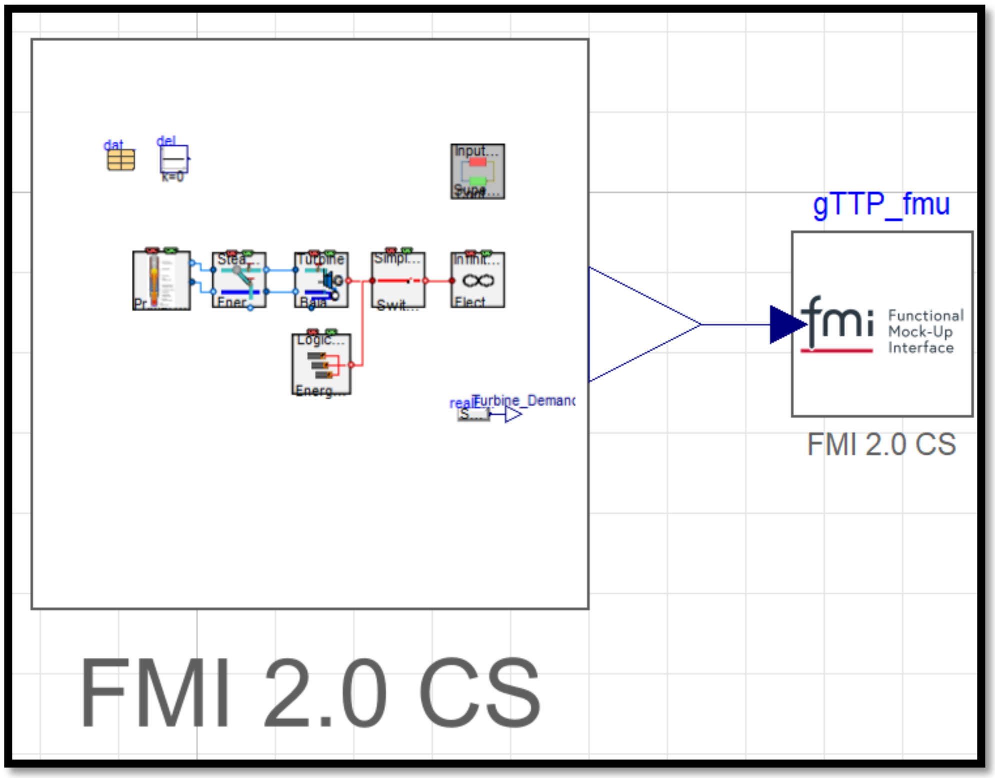


Figure 16. Integrated energy park FMI/FMU, including the Dymola GTTP model.

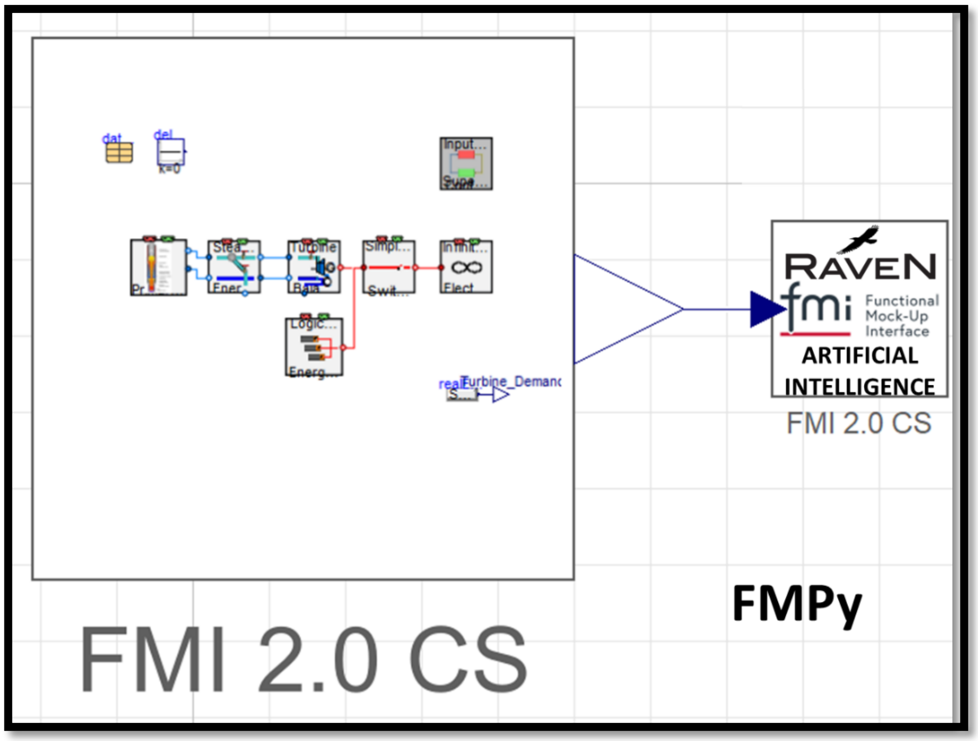


Figure 17. Integrated energy park FMI/FMU, replacing the Dymola GTTP model with the RAVEN AI-based FMI/FMU.

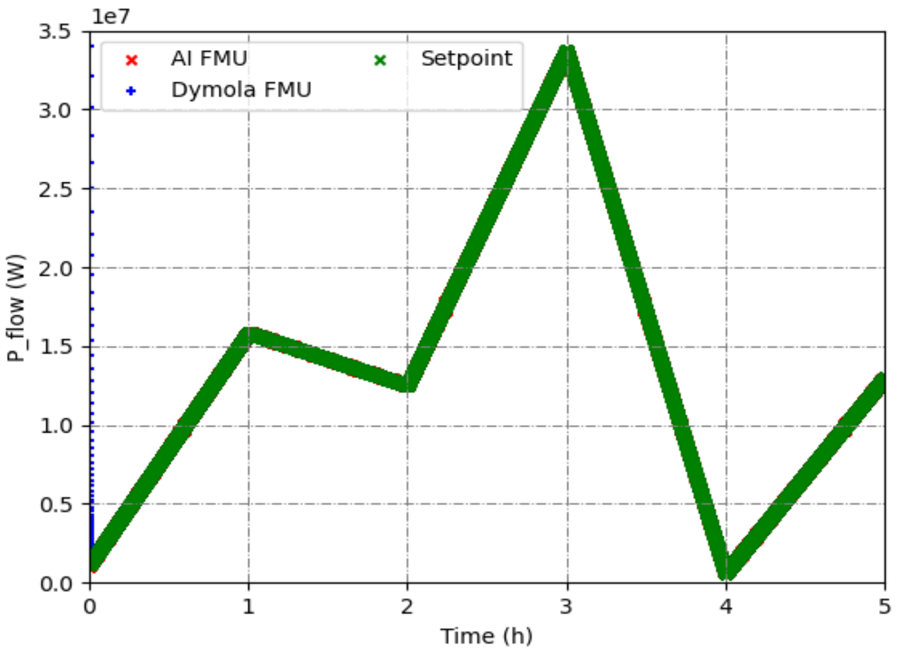


Figure 18. Co-Simulation FMI/FMU (Dymola) vs. RAVEN AI-based FMI/FMU for a 5-hour simulation of the turbine power vs. setpoint demand for the integrated energy park.

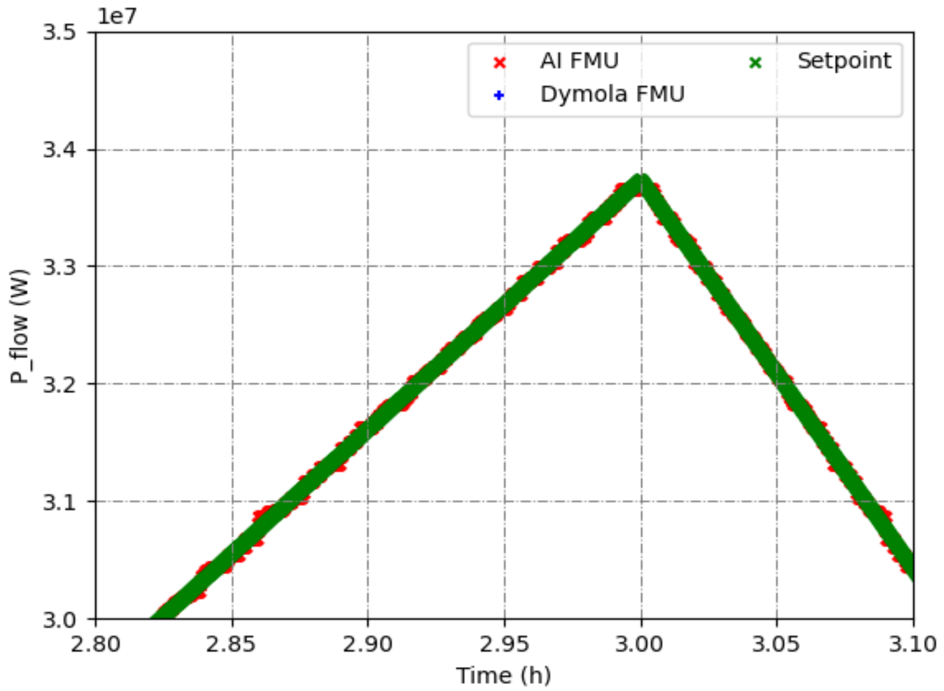
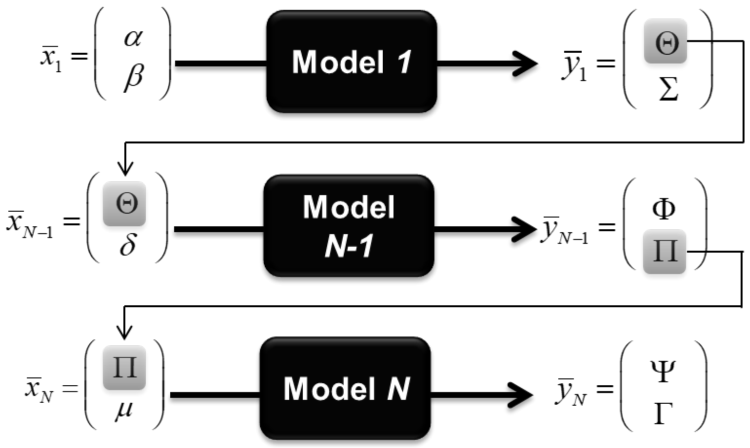


Figure 19. Co-Simulation FMI/FMU (Dymola) vs. RAVEN AI-based FMI/FMU closeup shot of turbine demand vs turbine output.

5. CONCLUSION

This report discussed the status of the flexible plug-and-play framework development currently ongoing that aims to integrate Modelica and Dymola with RAVEN in terms of FMI/FMU construction.

The report focuses on reporting some additional details regarding the work that was deployed to simulate, export, and use FMI/FMU in conjunction with AI algorithms in RAVEN represents a significant step forward regarding delivering a streamlined process to accelerate simulations and analysis by leveraging RAVEN advanced algorithms. The possibility of using AI exported in FMI/FMU in any FMI/FMU-compatible framework (e.g., FMPy and Dymola) is unique to this framework, posing the basis for deployment of fast simulation, modeling, and analysis accelerations.



$\bar{x}_1 = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$
Model 1
$\bar{y}_1 = \begin{pmatrix} \Theta \\ \Sigma \end{pmatrix}$
$\bar{x}_{N-1} = \begin{pmatrix} \Theta \\ \delta \end{pmatrix}$
Model N-1
$\bar{y}_{N-1} = \begin{pmatrix} \Phi \\ \Pi \end{pmatrix}$
$\bar{x}_N = \begin{pmatrix} \Pi \\ \mu \end{pmatrix}$
Model N
$\bar{y}_N = \begin{pmatrix} \Psi \\ \Gamma \end{pmatrix}$

Figure 20. RAVEN EnsembleModel capability (scheme).

6. FUTURE WORK

Over the next year, several tasks are planned to be carried out in the future of the program:

1) Master Simulator development in RAVEN: to automate the deployment of models in a system that is compatible with any FMI/FMU interface, an entity (Master Simulator) needs to be developed within RAVEN. Such development will allow for the simulation of FMI/FMU models (AI, Dymola, etc.) directly within the RAVEN framework allowing for the integration of such models in any RAVEN workflow, in general, and in IES technoeconomic analysis. The Master Simulator in RAVEN will be based on the EnsembleModel entity (see Figure 20), in conjunction with the FMPy library. The Master Simulator is shown in Figure 21.

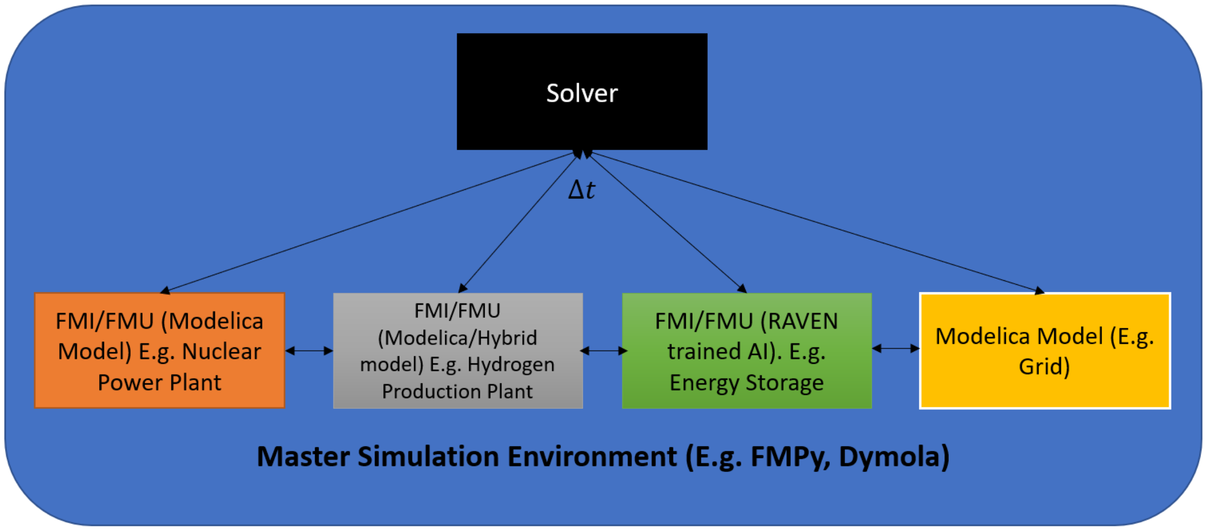

Solver
Δt
FMI/FMU (Modelica Model) E.g. Nuclear Power Plant
FMI/FMU (Modelica/Hybrid model) E.g. Hydrogen Production Plant
FMI/FMU (RAVEN trained AI). E.g. Energy Storage
Modelica Model (E.g. Grid)
Master Simulation Environment (E.g. FMPy, Dymola)

Figure 21. Proposed master simulator within RAVEN.

2) Model Exchange for RAVEN-based models: in section 3.2 the deployment of a system for exporting RAVEN-based AI as FMI/FMU leveraging, the PythonFMU library has been shown. However, the current library only supports FMI/FMU in co-simulation, useful for loosely coupled models but inadequate for tightly coupled systems. To allow for exporting of nonlinear models (e.g. Nuclear Reactor Balance of Plant, Storage, etc.), the PythonFMU library needs to be upgraded to allow for exporting models in model exchange and, consequentially, leverage the capability of RAVEN AI to provide first and second order derivative information.

3) Acceleration of FMI/FMU-based analysis using AI-High-Fidelity RAVEN scheme: RAVEN provides techniques for the acceleration of analysis using automated switch between AI and High-Fidelity models: the *HybridModel* entity. the *HybridModel* is a special class of model that is aimed to couple in-tandem, high-fidelity physical and mathematical models (e.g., FMI/FMU Dymola models) and artificial intelligence algorithms (e.g., surrogate models). The AIs are trained based on the results from the high-fidelity model. The global accuracy of the AIs is evaluated based on cross-validation scores, and the local (e.g., prediction) validity is determined via some local validation metrics (e.g., crowding distance). Once the AIs are trained, the *HybridModel* can decide which of the models (i.e., the AIs or high-fidelity model) to execute based on the previously mentioned accuracy and validation metrics. Figure 22 shows the scheme behind the hybrid model formulation. The future plan is to use this capability in techno-economic analysis to accelerate the process without losing accuracy.

4) Begin the creation of a graphical user interface for FORCE to enable external users (universities, analysts, industrial partners) to more readily create and test various configurations of integrated energy systems. This interface is envisioned to be able to create either a technoeconomic analysis work flow with HERON and RAVEN or a transient process model workflow using FMI/FMUs and Modelica models.

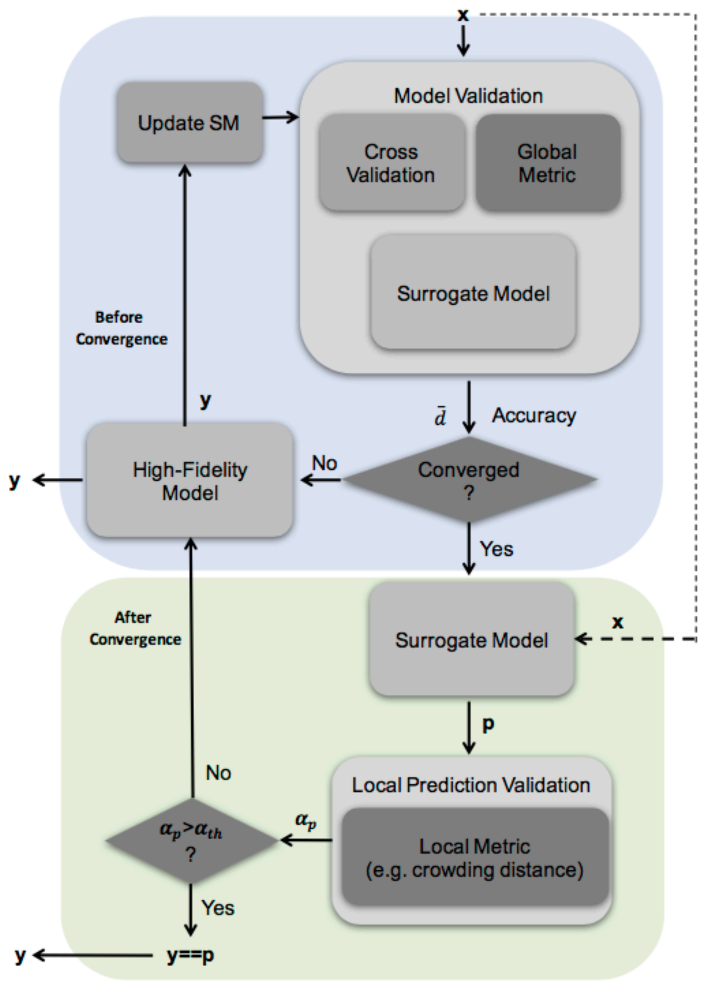


Figure 22. RAVEN hybrid model scheme.

7. REFERENCES

[1] Dassault Systems. “DYMOLA Systems Engineering: Multi-Engineering Modeling and Simulation Based on Modelica and FMI.” Accessed July 24, 2020. https://www.3ds.com/products-services/catia/products/dymola/.

[2] Alfonsi, A., C. Rabiti, D. Mandelli, J. Cogliati, C. Wang, P. W. Talbot, D. P. Maljovec, and C. Smith. 2016. “RAVEN Theory Manual and User Guide.” INL/EXT-16-38178, Idaho National Laboratory.

[3] Rabiti, C., A. Alfonsi, J. Cogliati, D. Mandelli, R. Kinoshita, S. Sen, C. Wang, and J. Chen. 2017. “RAVEN User Manual.” INL/EXT-15-34123, Idaho National Laboratory.

[4] Alfonsi, A., K. Frick, S. Greenwood, and C. Rabiti. 2020. “Status on the Development of the Infrastructure for a Flexible Modelica/RAVEN Framework for IES.” INL/EXT-20-00160, Rev 00, Idaho National Laboratory.

[5] Frick, K., A. Alfonsi, C. Rabiti. 2020. “Flexible Modelica/RAVEN Framework for IES.” INL/EXT-20-00419, Rev 00, Idaho National Laboratory.

[6] Frick, K., Alfonsi A., Rabiti C., Bragg-Sitton S., 2021. “Development of the IES Plug-and-Play Framework”. INL/EXT-21-62050. Rev 00, Idaho National Laboratory.

[7] FMpy. 2020. “FMpy 0.2.21.” Accessed July 8, 2020. https://pypi.org/project/FMPy/.

[8] PyFMI. 2018. “PyFMI 2.5.” Accessed March 25, 2020. https://pypi.org/project/PyFMI/.