Using Simulink with Python

Weiwu Li, MathWorks
Yann Debray, MathWorks
You probably have heard a lot about using MATLAB and Python
But wondered what about using Python with **Simulink**?
Key takeaways

- Simulink as an open simulation platform supports versatile ways to interoperate with Python:
  - Bring Python code into Simulink as a library for co-execution
  - Integrate TensorFlow and PyTorch models for both simulation and code generation
  - Simulate a Simulink model directly from Python
  - Export a Simulink model as a Python package for deployment
Why use Python and Simulink together?

- Need to integrate code from a colleague

- Facilitate development by enhancing an AI workflow

- Need functionality available in MATLAB and Simulink or in Python

- Leverage the work from the community
The best way to use Simulink with Python is case specific

- Let’s illustrate that through 4 typical scenarios

- In a team setting:
  - Weiwu is a Simulink user and Yann is a Python user
  - Yann and Weiwu need to work with each other to deliver a project
Scenario #1

I’m an algorithm developer using Python; I develop image processing and computer vision algorithms (and many more).

I’m a system engineer who integrates multiple components together. I want to simulate the whole virtual system including Yann’s Python algorithm in Simulink.
Calling Python from Simulink

- Use Python Importer

- Graphical wizard for step-by-step guidance, no/minimal manual code
- Integrating a package of Python functions with each Python function corresponding to a library block
- Convenient for re-use or building a custom blockset

Use **Python Importer** to bring Python functions into Simulink
Calling Python from Simulink
- Use Python Importer

- Demo: Integrate human detection algorithm in Python
Calling Python from Simulink  
- Using MATLAB Interface for Python

If you are using a version earlier than R2023a, or you would like to write code manually:

- Write MATLAB functions or MATLAB System objects
- Access Python libraries directly by adding the `py.` prefix to the Python name
Scenario #2

I’m a data scientist using TensorFlow & PyTorch to develop deep learning models (e.g., Battery State of Charge estimation).

Yann

I need to bring Yann’s pretrained deep learning model into Simulink for system validation. But **co-simulation is not enough, we also need code generation for hardware implementation.**

Weiwu
There are two options to bring your deep learning model into Simulink if code gen is a must have

- Import your deep learning model in MATLAB directly
  - Multi-platform code generation: library-free C/C++ code, optimized code for Intel and ARM processors, and CUDA code for NVIDIA® GPU
  - Import process can be painful, need for custom code, and validation testing

- Simulate and generate code for TensorFlow™ Lite model
  - Requiring only a simple Python code to compile the model
  - Requires the TensorFlow Lite interpreter and libraries built on the target hardware, which is currently limited to Windows and Linux targets
Integrate deep learning models from Python
-using MATLAB model converters for TensorFlow, PyTorch, and ONNX

• Once the model is converted in MATLAB, use the deep neural networks blocks to bring it into Simulink, for both simulation and code generation
Integrate deep learning models from Python

- using MATLAB model converters for TensorFlow, PyTorch, and ONNX

- Demo: Integrate a TensorFlow model for battery SoC estimation
Integrate deep learning models from TensorFlow Lite (TFLite) - using the MATLAB function to load a pre-trained TFLite model in Simulink.

Example: TFLite Object Detector integrated with Simulink

Import TFLite models using a MATLAB Function block.
Scenario #3

I use Simulink to model a dynamic system, for example, a vehicle suspension system.

Weiwu

I want to use a Python based automation framework to run Simulink simulations. I need to **invoke Weiwu’s Simulink model from Python** for automated testing.

Yann
Simulate a Simulink Model from Python
- Using MATLAB Engine API

```python
mle = matlab.engine.start_matlab();  # start the MATLAB engine
res[0] = mle.sim_the_model();  # # run Simulink simulation within a MATLAB function
```
Simulate a Simulink Model from Python
- Using MATLAB Engine API

- Create/terminate MATLAB

- Put variable into MATLAB workspace

- Get variable from MATLAB workspace

- Provide flexible Simulink simulation capabilities including changing non-tunable parameters and running simulations in normal mode
Simulate a Simulink Model from Python
- Using MATLAB Engine API

- Demo: Simulate a road suspension model in Python
Scenario #4

We need to deploy Weiwu’s Simulink model in a Python-based production environment. I want to get a Python package which encapsulates a Simulink simulation which can be used for deployment.

Yann
Call a compiled Simulink model from Python
- Using MATLAB Runtime

Generate a Python package from a MATLAB function that encapsulates a Simulink simulation

```
import sim_the_model
mlr = sim_the_model.initialize()
res[0] = mlr.sim_the_model()
```
Call a compiled Simulink model from Python
- Using MATLAB Runtime

- Demo: Simulate the compiled suspension system model as a Python package
Other ways to call a compiled Simulink model in Python

- Package the Simulink model as a Functional Mockup Unit (FMU)
  - Call the FMU from Python using third-party libraries such as FMPy

- Package the Simulink model as a simulation service API (using MATLAB Production Server)
  - RESTful API for scalable applications

- Generate C/C++ code or shared library from the Simulink model
  - Call the generated code using ctypes or related wrappers
Key takeaways

- Simulink as an open simulation platform supports versatile ways to interoperate with Python:
  - Bring Python code into Simulink as a library for co-execution
  - Integrate TensorFlow and PyTorch models for both simulation and code generation
  - Simulate a Simulink model directly from Python
  - Export a Simulink model as a Python package for deployment
Customer Reference: Mercedes-Benz Simulates Hardware Sensors with Deep Neural Networks

Challenge
Simulate automotive hardware sensors with deep neural networks

Solution
Use MATLAB, Simulink, Deep Learning Toolbox, and Fixed-Point Designer to convert Qkeras deep learning models into code that can be deployed to an automotive ECU

Results
- CPU, memory, and performance requirements met
- Flexible process established
- Development speed increased 600%

“This was the first time we were simulating sensors with neural networks on one of our powertrain ECUs. Without MATLAB and Simulink, we would have to use a tedious manual coding process that was very slow and error-prone.”
- Katja Deuschl, AI developer at Mercedes-Benz

Link to user story
To learn more

- Import Python Code to Simulink Using Python Importer Wizard
- Integrate Python Code with Simulink
- Deep learning with Simulink
- Importing Models from TensorFlow, PyTorch, and ONNX
- MATLAB Engine API
- Call Simulink from Python
- Python Package Integration
Thank you