Python-Modelica Framework for Automated Simulation and Optimization

Mareike Leimeister[1,2]

[1]Naval Architecture, Ocean and Marine Engineering, University of Strathclyde, UK
[2]Fraunhofer IWES, Fraunhofer Institute for Wind Energy Systems, Germany,
mareike.leimeister@iwes.fraunhofer.de

Modeling and simulation are essential for the development of complex engineering systems, such as wind turbines. Thus, Fraunhofer IWES has developed the MoWiT (Modelica for Wind Turbines) library for fully-coupled aero-hydro-servo-elastic simulations of wind turbine systems. To meet the needs for detailed assessment and design development of such sophisticated systems, which imply iterative steps for design optimization, a Python-Modelica framework is set up. By means of this, the simulation of MoWiT models can easily be managed, including redefinition of model parameters, specification of output sensors and simulation settings, integration of optimization algorithms, post-processing of simulation results, as well as parallel execution of several simulations. The application of this Python-Modelica framework is shown based on the example of a design optimization task of a floating wind turbine support structure.

The framework for automated simulation of wind turbine models requires a modeling environment (the MoWiT library); a tool for executing the time-domain simulations (Dymola); and a programming interface (Python) for external and automated control of the simulations. Figure 1 schematically represents the simulation framework and the working levels in Python.

This Python-Modelica framework for automated simulations serves as basis for further applications, such as the realization of optimizations. The pre-processed wind turbine system model, as well as definitions regarding the optimization process, are passed to the main script, by which means finally the execution of the optimization algorithm (Figure 2) is started. Additional information on the optimization is provided by separate classes, clustered into the optimization problem (comprising definitions of design variables, objective functions, as well as constraints), the optimizer, and the optimization algorithm.

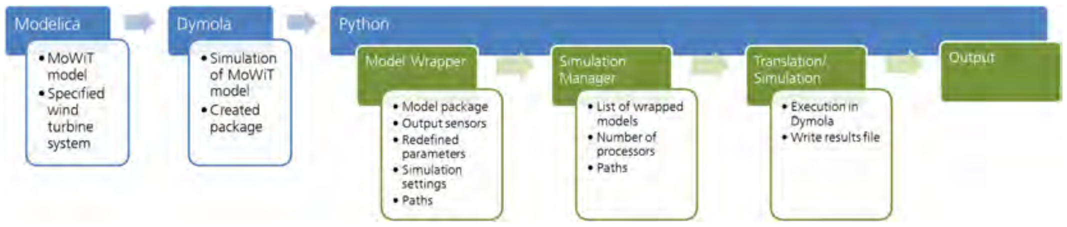

Figure 1. Simulation framework in Python.

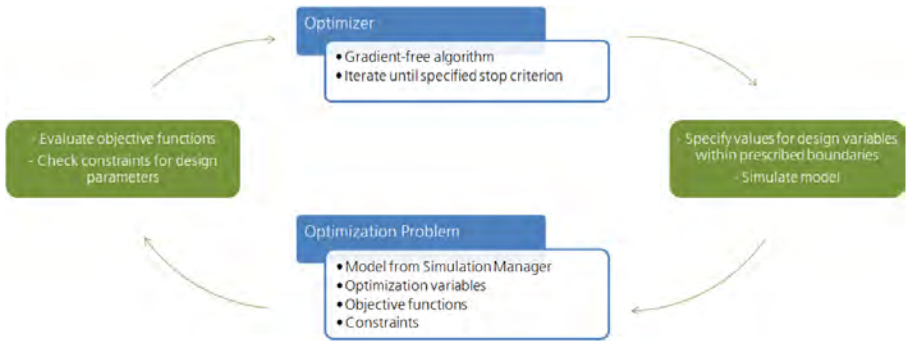
Optimizer
• Gradient-free algorithm
• Iterate until specified stop criterion
- Specify values for design variables within prescribed boundaries
- Simulate model
Optimization Problem
• Model from Simulation Manager
• Optimization variables
• Objective functions
• Constraints
- Evaluate objective functions
- Check constraints for design parameters

Figure 2. Automated optimization algorithm in Python.