

CPS: Praktikum 4

Simulator für Model Predictive Control

Version: 2025-11-30



This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Germany License. To view a copy, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/de/> or write to Creative Commons, 171 Second Street, Suite 300, San Francisco, CA 94105, USA.

Der vorbereitete Code für dieses Praktikum ist ein mittels Vibe-Coding entstandenes, aus sieben Python-Dateien bestehendes Programm. Es simuliert Model Predictive Control für eine aus Wärmepumpe, Photovoltaik und Batteriespeicher bestehende Anlage. Die Herausforderung: Es stecken logische und fachliche Fehler im Programm.

Der `ScenarioGenerator` soll den Zeitverlauf von Außentemperaturen, Globalstrahlungswerten, Strompreisen und einer Basislast im Haus erfinden. (In einer Forschungsarbeit würde man hierfür historische Zeitreihen benutzen.) Der `ForecastGenerator` erzeugt daraus Vorhersagen, die über den Verlauf des Zeithorizonts zunehmend unsicherer werden. Das Modell erhält diese Werte als externe Störungen: `Disturbance`.

Das `BaseModel` modelliert das Verhalten des Systems über einen Zeitschritt. Es wird als Modell im MPC benutzt, dient aber obendrein bequemlichkeitshalber in der Simulation als Modell des realen Systems, um die Ist-Werte zu erhalten. Diese zweite Anwendung des Modells als `RealPlant` hat leicht andere Parameter, um zumindest ein wenig zu demonstrieren, dass MPC-Modell und Realität (hier: die zweite Anwendung des Modells) nicht identisch sind. Außerdem liefert `RealPlant` zufällig gestörte Messwerte.

Die Zielfunktion ist in `Objective` einprogrammiert. Die Zielfunktion, das Modell und die Vorhersagen werden von `GeneticMPCController` benutzt, um einen möglichst optimalen Verlauf der Stellgrößen zu finden. Der Optimierer darin verwendet ein genetisches Verfahren: Eine Population verschiedener Verläufe wird über mehrere Generationen gemixt und mutiert. Das ist zwar langsam, aber allgemein anwendbar und noch relativ einfach zu verstehen. Tipp: Zum schnellen Ausprobieren von Änderungen reduzieren Sie die Größe der Population und/oder die Zahl der Generationen.

Aufgaben:

- Studieren Sie die am Ende des Programms erzeugten Diagramme, suchen Sie darin nach Fehlern und beseitigen Sie die entsprechenden logischen/fachlichen Bugs im Programm.
- Überprüfen Sie die Größenordnungen der von der KI automatisch gewählten Konstanten für die Modelle, die Zielfunktion usw.
- Überprüfen Sie, ob die Optimierung funktioniert. Und was sind effiziente Parameterwerte für die genetische Optimierung (Zahl der Generationen usw.)?
- Berücksichtigen Sie eine feste Einspeisevergütung.

-
- Überlegen Sie sich verschiedene Ansätze dafür, dass der Verlauf der Leistung der Wärmepumpe glatt wird und das auch im Modell berücksichtigt wird. Implementieren Sie einen davon.
 - Die Vorhersage könnte neben dem erwarteten Wert auch dessen Unsicherheit angeben. Alternativ könnte man ein Ensemble mehrerer Vorhersagen haben. Wie würden Sie solche zusätzlichen Daten berücksichtigen?
 - Seit Version 3.14 kann Python offiziell alle Kerne des Prozessors parallel nutzen (derzeit aber erst in besonderen Python-Builds mit Namen wie 3.14t, 3.14-freethreading). Was könnte man damit in dieser Simulation beschleunigen? Wie?