PyMoskito

PyMoskito steht für **Py**thon based **Mo**dular **S**imulation & Postprocessing **Ki**ckass **To**olbox und kann als universelles Werkzeug zur Simulation von typischen Problemen in der Regelungstechnik und anderen Bereichen eingesetzt werden. Die stärken liegen bei der Simulation von vielen Konfigurationen mit diversen Variationen in der Struktur oder in den Parametern.

Der Grundstein für PyMoskito wurde im Wintersemester 2014/2015 in Form eines Oberseminars zum Thema „Untersuchung der Ball and Beam Problematik“ gelegt. Dieses Oberseminar beinhaltete unter anderem die Aufgabe das System mit Python zu simulieren. Daraus entwickelte sich Stück für Stück die hier vorliegende Toolbox.

Zur Übersicht und einfachen Bedienung der Simulation enthält PyMoskito eine Benutzeroberfläche (GUI). Mit Hilfe der GUI werden folgende Funktionen abgedeckt:

- einfache Konfiguaration des Regelkreises
- Simulation einer eingestellten Regelkreisstruktur
- Laden von gespeicherten Simulationskonfigurationen
- automatische Simulation von Simulationssätzen
- Speichern von Simulationsergebnisse zur späteren Bearbeitung
- Schnittstelle zur 3D-Visualisierung des physikalischen Systems mit Hilfe von VTK
- Abspielfunktionen für die 3D-Visualisierung
- Darstellen der Simulationsergebnisse durch Graphen
- Möglichkeit zur individuellen Weiterbearbeitung von Simulationsergebnissen im Postprocessing-Fenster
- Verschiebbarkeit und Skalierung der einzelnen Fenster

PyMoskito bietet einen Simulationskern zur Ausführung der in Abbildung 1 dargestellten Regelkreisstruktur und eine Nachbearbeitung von Simulationsergebnissen an. Die Module dieses Regelkreises sind hierbei einzeln konfigurierbar und bieten so die Möglichkeit verschiedenste Szenarien gleichzeitig abzudecken. Sie unterteilen sich dabei in zwei verschiedene Gruppen:

I. generische, daher vom konkreten System unabhängige Blöcke die damit auch für andere Systeme Verwendung finden können und

II. systemabhängige, speziell zugeschnittene Blöcke, die für jedes System neu implementiert werden müssen.

Das System ist Modular aufgebaut, sodass Blöcke ausgetauscht oder weggelassen werden können. Neben dem eigentlichen Simulationssteuerung werden generische Klassen zur Verfügung gestellt, welche häufig benutzte Simulationselemente enthalten. Dazu zählen:

- ODEInt, ein Integrator zur Lösung von Differentialgleichungssystemen 1. Ordnung

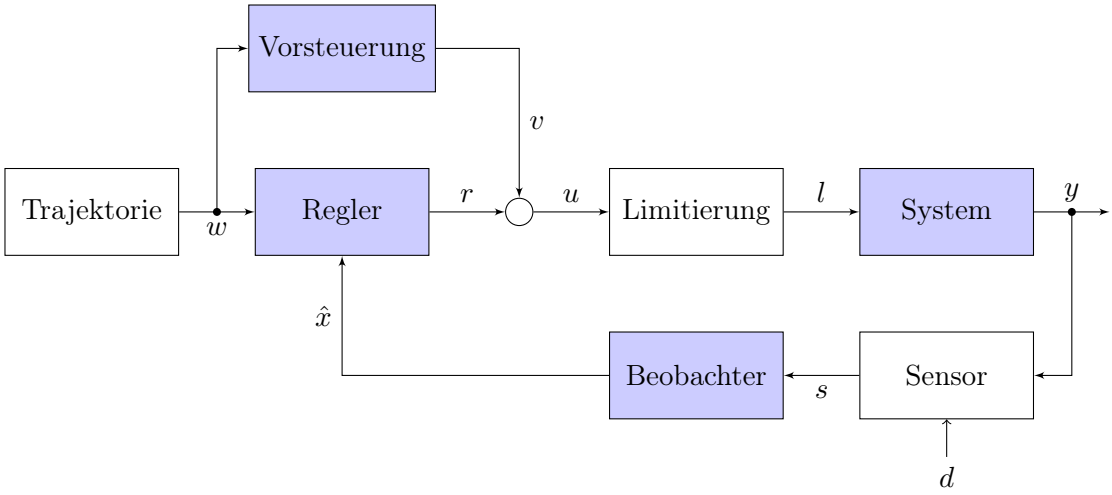

Abbildung 1: Schematische Darstellung des Regelkreises. Blau hinterlegte Komponenten sind an das jeweilige System anzupassen, alle anderen Blöcke sind generisch verwendbar.

- SmoothTransition, ein Trajektoriengenerator zur Erzeugung von glatten Übergängen zwischen gegebenen Zuständen und Zeitpunkten
- HarmonicTrajectory, ein Trajektoriengenerator zur Erzeugung von harmonischen Sollwerten mit gegebener Frequenz und Amplitude
- PIDController, ein Standard Regler in der Regelungstechnik
- Limitierung (noch zu implementieren)
- Sensor (noch zu implementieren)

In PyMoskito gibt es zur Orientierung und als Hilfe bei der Implementierung einer eigenen Problemstellung, ein Beispiel. Es handelt sich um das in der Regelungstechnik beliebte „Ball and Beam“ System. Bei der „Ball and Beam“ Problematik geht es darum, einen Ball auf einem in der Mitte drehbar gelagerten Balken auf eine bestimmte Position zu regeln. Aus diesem Beispiel ist auch ersichtlich, welche Module bzw. Dateien bei einer neuen Problemstellung zu erstellen sind. Dazu gibt es in PyMoskito verschiedene Basisklassen, die überladen werden müssen. Dies stellt sicher das alle für die Simulation benötigten Bestandteile implementiert werden. Die folgende Aufstellung zeigt, die selber zu entwerfenden Elemente:

- Zustandsraummodell ist entsprechenden dem Beispiel als right hand side zu hinterlegen
- in der auszuführenden Datei muss die Benutzeroberfläche erstellt und initialisiert werden
- Regelgesetze zur Stabilisierung des Regelkreises
- Entwurf einer Standardsimulationskonfiguration
- Optional: 3D-Visualisierung mit Hilfe von VTK