3DCSM - Body Handling

Simulations-Code Dokumentation
Code-Struktur und Simulationsablauf

Bernd Heufelder

23. April 2019

Inhaltsverzeichnis

1 Bewegungsgleichung erstellen

Die komplette Erzeugung der Bewegungsgleichungen erfolgt im File *eq_of_motion.py*. Die Bewegungsgleichungen werden als Binärfiles im Format *.pickle* abgespeichert. Die benötigten Parameter werden aus dem File *param.py* importiert.

2 Simulation des Modells

Gestartet wird die Simulation über das File simulation.py durch die Klasse SIMULATION.

oSim = Simulation(): In der *__init__()* wird zuerst das Modell *model_full()* instanziert, dann ein Trajektorienobjekt entweder anhand von externen Textfiles oder durch Vorgabe von Start- und Endwerten erzeugt. Die Zustandsvariablen zum Zeitpunkt Null werden anhand des Trajektorienobjektes gesetzt.

1. ***model_full.__init__():*** Dieses wiederum instanziert das mechanische und elektrische Modell. Dafür werden die Files *model_full.py, model_mech.py, model_el.py* verwendet.

   model_mech.__init__(): Hier werden die Binärfiles der Bewegungsgleichungen geladen und in dieser Klasse als Properties gespeichert. Zusätzlich wird als Property des mechanischen Modells der PID-Regler der Trajektorie aus dem File *control.py* instanziert.

   PID_traj.__init__(): Hier werden die Regelparameter für den Trajektorienregler als Properties der PID_TRAJ Klasse definiert.

   model_el.__init__(): Hier wird nur der Stromregler der Motoren über die Klasse PID_MOTOR instanziert.

   PID_motor.__init__(): Hier werden die Regelparameter für den Trajektorienregler als Properties der PID_MOTOR Klasse definiert.

2. ***Trajektorienobjekt erstellen***

3. ***Initialwerte des Modells aus der Trajektorie setzen***

4. ***Arrays zur Datenspeicherung erstellen***

oSim.integrate(): Hier werden die Bewegungsgleichungen anhand eines ODE-Solvers von den Initialwerten aus zum Zeitpunkt Null bis zum Endzeitpunkt iterativ gelöst. Dem Solver wird als Systemfunktion die Funktion *system_equations(t, state, oTraj)* aus dem File *model_full.py* zugewiesen. Dem Solver kann eine Schrittweite *dt* angegeben werden, welche die zeitliche Differenz zwischen zwei Integrationsschritten vorgibt. Der Solver hat aber ebenso eine Variable *nsteps*, welche die maximale Anzahl an Zwischenintegrationsschritten vorgibt. Das heißt, der Solver integriert innerhalb eines $dt - Intervalls$ maximal nochmals *nsteps*-mal um eine gewünschte Toleranz des Ergebnisses zu gewährleisten.

oSim.save_results():

oSim.analyse_mech_eigenfreq():

3 Ergebnisse darstellen und visualisieren