Inbetriebnahme und Modellierung einer Asynchronmaschine

Lukas Tetzlaff, Robby Kozok, Nic Fränky Siebenborn und Pascal Kahlert

***Abstract*—Dieser Anwendungshinweis beschäftigt sich mit der einfachen Inbetriebnahme einer Asynchronmaschine vom Typ MDFMAIG071-12D von Lenze. Außerdem wird in diesem Dokument die Benutzung eines passendes Models behandelt.**

***Index Terms*—IEEE, IEEEtran, journal, LaTeX, paper, template.**

I. Introduction

HIS

Einleitung schreiben

IEEEtran.cls version 1.8b and later. I wish you the best of success.

Nic Siebenborn, Robby Kozok, Lukas Tetzlaff, Pascal Kahlert

17 Februar, 2016

II. Modellierung der Asynchronmaschine

Dieser Abschnitt befasst sich mit der Benutzung der beigefügten Simulation. Es soll beschrieben werden welche Parameter angepasst werden können. Es sollen weiterhin verschiedene Simulationen dargestellt werden um darzustellen wie sich ein richtig eingestellter Motor verhalten sollte. Die Modellierung wurde in drei Abschnitte unterteilt.

- Feldorientierte Regelung
- Transformation
- Motormodell

A. Konfiguration des Motormodells

Dieses Modell kann grundsätzlich an jeden Asynchronmotor angepasst werden. Es muss dabei nur beachtet werden, dass alle Parameter im *Model Explorer* richtig eingestellt werden. Es folgt nun eine Auflistung von Motorparametern, welche bekannt sein müssen.

Bezeichnung	Beschreibung	*Model Explorer*
Elektrische Kenngrößen		
I_μ^+	Magnetisierungsstrom	I_magnet
Ψ_R^+	Läuferfluss	PsiR_plus
L_R^+	Läuferinduktivität	I_magnet
L_σ^+	Streuinduktivität	Lsigma_plus
R_R^+	Läuferwiderstand	RR_plus
R_S	Ständerwiderstand	Rs
I_A	Start-Strom	StartStrom
M_A	Start-Moment	StartMoment
p	Polpaaranzahl	p
Mechanische Kenngrößen		
μ	Reibungskoeffizient	Rreib
J	Trägheitsmoment	J

Diese Parameter müssen in den *Model Explorer* eingetragen werden. Fehlt einer oder mehrere Parameter kann es passieren, dass das Model kaum oder garnicht funktioniert, also sich nicht wie erwartet verhält.

Um nun eine Simulation sinnvoll durchzuführen, müssen noch Sollvorgaben und Lastsituationen eingetragen werden. Dafür können die die Blöcke Drezahlvorgabe und Lastvorgabe genutzt werden. Diese Blöcke verwenden die Parameter MLast und wSoll, diese sollten ebenfalls an realistische Last-Situationen angepasst werden.

1) Bestimmung von dem Reibungskoeffizienten μ: Sollte μ nicht bekannt sein, lässt sich dieser leicht über die Simulation ermitteln. Dafür sind folgende Schritte notwendig.

a) Anpassung der Simulation: Im ersten Schritt muss der Momentregler vom Rest der Simulation getrennt werden und durch das konstante Nennmoment ersetzt werden, dies ist in in **??** zu sehen.

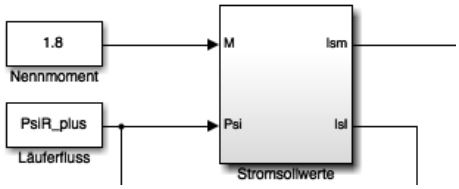

Fig. 1. Aufbau der Simulation zur Bestimmung des Reibkoeffizient

b) Bestimmung des Parameters: Nun muss μ solange verändert werden, bis die Drehzahl gleich der Nenndrehzahl wird. Dabei müssen alle MLast gleich null sein. Dies ist im *Model Explorer* einzustellen. In **??** ist Beispielhaft zu erkennen, wie sich die Drehzahl der Nenndrehzahl nähert.

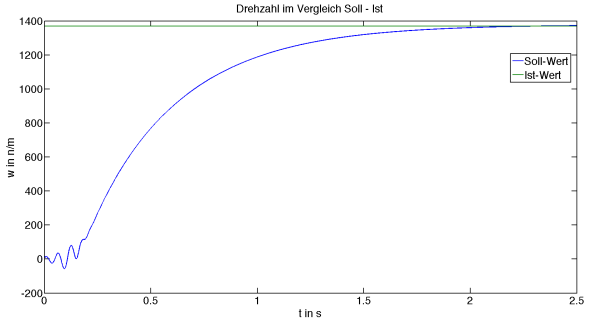

Fig. 2. Zeitlicher Verlauf der Drehzahl im Vergleich zur Nenndrehzahl(hier Solldrehzahl)

B. Simulation der Asynchronmaschine

Im folgenden sollen verschiedene Simulations-Szenarien dargestellt und erläutert werden.

1) Sollwertvorgabe - ohne Last: Dieses Szenario kann genutzt werden um die grundsätzliche Funktion des Modell sicherzustellen.

a) Einstellung der Parameter: Im ersten Schritt müssen alle MLast Parameter auf 0 gesetzt werden(siehe **??**). Außerdem sollten sinnvolle WSoll Vorgaben gemacht werden, so könnte zum Beispiel folgender Ablauf gewählt werden:

Zeitpunkt t	Solldrehzahl n_{Soll} in n/min
1s	500
6s	1370

b) Durchführung der Simulation: Im nächsten Schritt sollte eine sinnvolle Zeit für die Simulation gewählt werden. In diesem Beispiel haben wir 10s gewählt.

c) Analyse der Ergebnisse: In **??** ist zu sehen, dass der Motor auf Vorgabeänderungen der Drehzahl mit starken Schwankungen des Drehmomentes reagiert.
Bei einer Erhöhung der Drehzahl wird der Motor versuchen das Drehmoment innerhalb seiner Grenzen(Strom, Spannung, etc.) zu erhöhen um die gewünschte Drehzahl zu erreichen. Bei einer Überschreitung wird er ruckartig das Moment verringern. Bei einer Verringerung der Drehzahl reagiert der Motor damit, das Moment zu verringern, dort ist zu sehen wie der Moment des Motors negativ wird, da dem Motor fast keine Energie mehr zugeführt wird und damit auch kein Antrieb im Motor steckt.

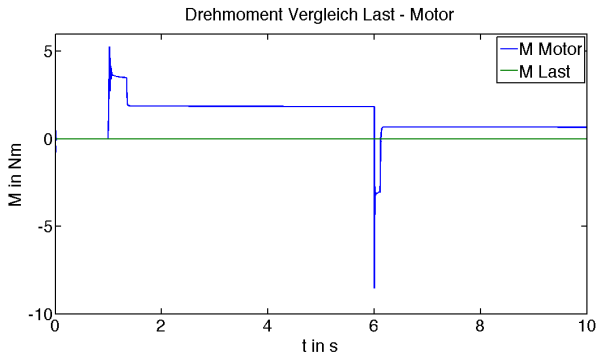

Fig. 3. Zeitlicher Verlauf des Drehmomentes

In **??** sind keine Besonderheiten zu erkennen, der Motor versucht nach seinen Möglichkeiten der Vorgabe zu folgen und schafft dies in einer relativ kurzen Zeit.

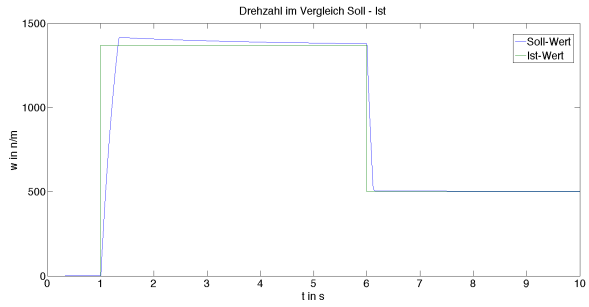

Fig. 4. Zeitlicher Verlauf der Drehzahl

2) Sollwertvorgabe - mit Last: Diese Simulation zeigt das Lastverhalten des Motors. Es soll gezeigt werden, dass dieser bei seinem angegebenen Haltemoment stehen bleibt und bei zu großer Last sogar rückwärts dreht.

a) Einstellung der Parameter: Für diese Simulation sollten sinnvolle Werte gewählt werden. In unserem Fall wurden die Werte so gewählt, dass im ersten Zeitfenster(1s-6s) zu sehen ist wie die Maschine durch eine Last gehalten wird und im zweiten Zeitfenster(6s-10s) zu sehen ist wie die Last die Maschine dreht.

Zeitpunkt t	n_{Soll} in n/min	M in Nm
1s	1370	3,8
6s	500	10

b) Analyse der Ergebnisse: In **??** is zu sehen, dass das Drehmoment des Motors und der Drehmoment der Last sich angleichen, da der Motor versucht die eingestellte Drehzahl zu erreichen. Ab Sekunde 6 ist zu sehen, dass das Drehmoment des Motors deutlich unter dem der Last ist. Auch wenn der Motor im ersten Moment versucht, gegen die Last anzukommen bringt ihn dann die simulierte Begrenzung des Stromes wieder auf sein Haltemoment.

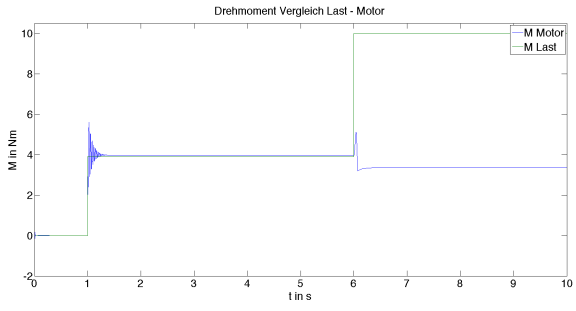

Fig. 5. Zeitlicher Verlauf des Drehmomentes

Einen passenden Verlauf können wir in **??** sehen. Im Zeitraum von 1s - 6s ist zu sehen, wie der Motor nahezu steht. Sobald dann ab Sekunde 6 das Moment der Last größer als sein Haltemoment wird, fängt der Motor an, sich rückwärts zu drehen.

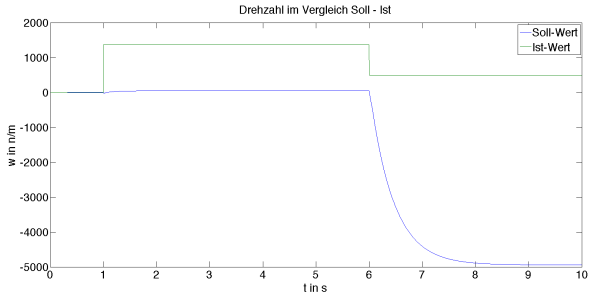

Fig. 6. Zeitlicher Verlauf der Drehzahl

C. Schwächen des Modells

Dieses Modell stellt keine reale Abbildung des Motors dar, kann aber einen Eindruck vermitteln, welches Verhalten von diesem Asynchronmotor zu erwarten ist.
Vor allem in der Begrenzung von Strom und Spannung ist deutliches Verbesserungspotential zu sehen.
So wird die Spannung im gesamten Modell nicht begrenzt und steigt teilweise auf bis zu 1500V, dies ist mit einer Versorgungsspannung von 230V-AC aber nicht möglich. Änderungen dieser Parameter können an den Reglerbausteinen vorgenommen werden, dies führt allerdings zu einer Neueinstellung der Parameter.
Es ist zu erwarten, dass der Motor in der Realität nicht so dynamisch reagiert wie im Modell, da eine Begrenzungen fehlen.

III. Inbetriebnahme der Asynchronmaschine

A. Verkabelung

Da der in diesem Projekt verwendete Motor, schon einmal von einer anderen Gruppe in Betrieb genommen wurde, mussten wir lediglich die schon verdrahteten Anschlusskabel, anschließen. Folgende Verbindungen müssen gesteckt werden:

- Die Leitungen U,V und W der Asynchronmaschine werden an das TI Kit geklemmt
- Die Leitungen des Drehimpulsgebers werden an den Anschluss ”QEP Connector” angeschlossen.
- Die 12V Spannungsversorgung wird an das TI Kit angeschlossen
- Das USB Kabel ist an das TI Kit und den PC anzuschliessen
- Die Kaltgerätestecker des TI Kits und des Lüfters sind zu stecken

B. Software-Realisierung

1) Erster Ansatz: Als ersten Ansatz wählten wir die naheliegende Idee die Exportfunktion von Matlab/Simulink Modellen zu legitimem C-Code zu nutzen. Hierbei ließ sich die C2000-Architektur auswählen und augenscheinlich korrekte Abbildungen der Simulationsblöcke wurden in einzelnen C-Sourcecodedateien erstellt. Die Blöcke wurden modular mit einem Eingangs- und Ausgangsstruct des jeweiligen, meist etwas wenig verbosen Typs (bspw. `ExtY_Koordinatentransfer1_T`, versehen, deren Member den Datenflüssen aus der Simulation entsprechen. Jedes dieser Modelle beinhaltete zudem eine Funktion `<Modellname>_step`, die einen Simulationsschritt darstellt und so bei uns nach Konfiguration des entsprechenden Schrittintervalls in einer Timerroutine benutzt worden wäre.

Final scheiterte der Ansatz jedoch daran, dass das elektrische Modell des Motors nicht nachgestellt und exportiert werden konnte und so der komplexeste Teil der Simulation noch immer übrig geblieben wäre. Diese kurze Ausführung soll daher gern als Abschreckung gesehen werden und zur Vorsicht in Bezug auf modellgenerierten Code für die C2000-Reihe aufrufen.

2) Tatsächliche Realisierung: Die tatsächlich genutzte Implementierung benutzt 6 PWMs aus dem TI-ePWM-Modul, durch deren Dutycycle-Veränderung über Zeit ein Pulsmuster erzeugt wird, mit dem die 6 MosFETs so beschaltet werden, dass der Motor sich dreht. Dies geschieht durch ein sukzessives Inkrementieren einer Zählvariablen in der ISR-Routine jedes der drei EPWM-Handlers (alle 0.4ms). Diese Zählvariable dient dann als Index für ein vorgeneriertes Array mit Sinuswerten, dezimal normiert auf +/- 1600 (2000 ist der eingestellte Registerwert, der die Periode darstellt, eine 1600 steht bspw. für einen Duty Cycle von 20% (1-(1600/2000))). Der Wert war bewusst nicht auf die volle Aussteuerung ausgelegt, um als Test einen gemäßigten Betrieb zu nutzen und Übermodulation beziehungsweise Blocktaktung zu vermeiden (zumindest im Betrieb ohne Regler, nachträglich wurde dann ein PI-Regler implementiert, durch dessen Regelabweichung es zu Blocktaktung kommen kann).

Beim übertragen der Werte aus dem Sinus-Array wird also der Reload-Wert des jeweiligen ePWM-Registers gesetzt, der damit die Zeit für beide logischen Zustände festlegt. Da es sich um 6 PWMs handelt, verhalten sich jeweils zwei zueinander entgegengesetzt und diese 3 Paare wiederum um 120 Grad verschoben zueinander.

3) Hinweise zum CCS-Projekt: Das Makefile des Demoprogramms, das wir als Framework nutzten, sucht teilweise fehlerhaft s. Bugreport an absoluten Pfaden, zum Beheben haben wir einen symbolischen Link auf das ”angebliche” Verzeichnis gesetzt:

```
mklink /D "C:\TI\controlSUITE2_DMC␣Rev""C:\
Beuth\ti_controlSUITE\"
```

Das Projekt nutzt zahlreiche Includes, teilweise aus DSP2803x_headers und DSP2803x_common, aber häufig auch aus den TI-Installationsverzeichnissen beziehungsweise wieder absolute Pfade, daher empfiehlt es sich wahrscheinlich meistens TI-Produkte nur mit Standardeinstellungen (gegebenenfalls in einer virtuellen Maschine) zu installieren.

References

[1]