Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Tillmann Baer

Software Tool zur Simulation und Visualisierung des stationären Betriebsverhaltens der Asynchronmaschine

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Tillmann Baer

Software Tool zur Simulation und Visualisierung des stationären Betriebsverhaltens der Asynchronmaschine

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Ing Michael Röther
Zweitgutachter : Prof. Dr. Ing Gustav Vaupel

Abgegeben am 6. August 2013

Tillmann Baer

Thema der Bachelorthesis

Software Tool zur Simulation und Visualisierung des stationären Betriebsverhaltens der Asynchronmaschine

Stichworte

Drehmoment-Drehzahl-Kennlinie, Kreisdiagramm, Zeigerdiagramm, Asynchronmaschine, Matlab GUIDE, Simulation, Software

Kurzzusammenfassung

Diese Bachelorthesis befasst sich mit der Entwicklung eines Programms mit einer graphischen Benutzeroberfläche zur Simulation des stationären Verhaltens von Asynchronmaschinen. Erstellt wird das Programm mit Matlab GUIDE. Es kann die charakteristischen Kennlinien simulieren und anschließend graphisch visualisieren. Ausgangspunkt für die Berechnungen ist das einphasige Ersatzschaltbild einer Asynchronmaschine.

Tillmann Baer

Title of the paper

Software tool to simulate and visualise the stationary operating behaviour of asynchronous machines

Keywords

Torque-speed, circle diagram, vector diagram, asynchronous machine, Matlab GUIDE, simulation, software

Abstract

This thesis is concerned with the development of a program with a graphical user interface to simulate and visualise the stationary operating behaviour of asynchronous machines by using Matlab GUIDE. The program can simulate and visualise the characteristic diagrams. The calculations are based on the single-phase equivalent circuite diagram of an asynchronous machine.

Danksagung

Danken möchte ich an dieser Stelle Prof. Dr. Röther für die Anregungen und Unterstützung während der Umsetzung dieser Bachelorthesis. Weiterhin danke ich Prof. Dr. Vaupel für die Zweitprüfung und Herr Korpel für die Bereitstellung der Laborausrüstung. Letztendlich möchte ich auch meinen Eltern, meiner Schwester und meinen Freunden danken, die mich während meines Studiums und während der Anfertigung dieser Arbeit immer unterstützt haben.

Inhaltsverzeichnis

1 Einleitung

Diese Bachelorthesis befasst sich mit der Entwicklung eines Software Tools zur Simulation von Asynchronmaschinen. Mit Hilfe dieses Tools soll es möglich sein, das stationäre Betriebsverhalten von Asynchronmaschinen nachzubilden und zu visualisieren. Es sollen die Drehmoment-Drehzahl-Kennlinie, das Kreisdiagramm sowie das Zeigerdiagramm simuliert und graphisch dargestellt werden können. Ausgangspunkt für die Berechnungen sind die Nenndaten der Asynchronmaschine, sowie die Parameter des einphasigen Ersatzschaltbildes (ESB).

Erstellt wird das Programm mit Hilfe von Matlab 2012b und dem dazugehörigen Tool GUIDE. Es soll über eine graphische Benutzeroberfläche verfügen und die Kennlinien anschaulich visualisieren. Das Programm soll später zum einen im Rahmen der Vorlesung Grundlagen der Energietechnik, welche im 4. Semester an der Hochschule für angewandte Wissenschaften (HAW) gehalten wird, und zum anderen im vorlesungsbegleitenden Laborversuch "GEP4 Drehstromasynchronmaschine" zur Erweiterung des Laborversuchs eingesetzt werden. Hierfür ist es wichtig, dass zum einen die Graphiken groß genug dargestellt werden können, um diese auch in der letzten Sitzreihe in einem Hörsaal erkennen zu können, und zum anderen die intuitive Bedienung, da während eines Laborversuchs kaum Zeit besteht, sich in eine komplexe Programmstruktur einzuarbeiten. Dadurch stehen bei der Programmierung die einfache Bedienung sowie eine anschauliche Visualisierung der simulierten Ergebnisse besonders im Fokus. Im Jahr 2012 hat es zu dieser Thematik eine Vorgängerthesis gegeben, deren Verbesserungsvorschläge aus dem Fazit teilweise mit aufgegriffen wurden [4][1]. Bei der Vorgängerthesis war vor allem die Visualisierung noch nicht ausgereift und dadurch der Einsatz für oben beschriebene Aufgabengebiete nur bedingt möglich. Daher galt es vor allem die Visualisierungen in der vorliegenden Bachelorthesis zu optimieren.

[1] Seite 45

2 Theoretische Grundlagen zur Asynchronmaschine

Die erste Asynchronmschine (ASM) wurde bereits 1889 von Michael Dolivo Dobrowolski entwickelt, nachdem um 1885 Nicola Tesla und Galileo Ferraris die Entstehung eines Drehfeldes durch mehrsträngige Wicklungen erforscht hatten [2][2]. Mittlerweile sind ASM die am häufigsten eingesetzten Motoren, da sie durch ihren einfachen und robusten Aufbau preisgünstiger und wartungsärmer sind als vergleichbare Gleichstrommaschinen. Die Einsatzgebiete der ASM sind breit gestreut und reichen von kleinen Leistungen bis 1kW für z.B. Haushaltsgeräte bis hin zu großen Antriebsmotoren von einigen MW für industrielle Antriebe.

Konstruktiv bestehen ASM immer aus einem feststehenden Ständer und einem Läufer, wobei man zwischen Käfig- und Schleifringläufern unterscheidet.

Die Stränge der Ständeranschlüsse werden an ein symmetrisches Drehstromnetz angeschlossen, so dass in jedem Strang ein um 120° phasenverschobener Strom mit identischer Amplitude und Frequenz fließt. Diese Ströme erzeugen ein magnetisches Drehfeld, welches dafür sorgt, dass Spannungen im Läufer induziert werden. Durch diese Spannungen werden Ströme im Läufer erzeugt, die dafür sorgen, dass sich ein magnetisches Drehfeld ausbildet und es zu einer Drehmomentbildung kommt [6][3].

2.1 Stationäres Betriebsverhalten der Asynchronmaschine

Um das Verhalten einer Asynchronmaschine zu berechnen und zu analysieren, benötigt man zunächst ein Ersatzschaltbild, welches die realen Verhältnisse möglichst genau wiederspiegelt.

Ausgangspunkte dieser Arbeit sind das einphasige ständerseitige Ersatzschaltbild (ESB) einer Asynchronmaschine und die Nomenklatur, welche Bestandteile des Vorlesungsskriptes sind (s. Abb. 2.1) [3][4]. Da in dem ESB nur ein Strangpaar (Läufer + Ständer) betrachtet wird, aus dem die beiden anderen Strangpaare abgeleitet werden, muss eine Symmetrie der Stränge in Bezug auf Aufbau und Belastung gegeben sein. Dies wird für die folgenden Berechnungen vorausgesetzt wird. Bei dem ständerseitigen Ersatzschaltbild sind die läuferseitigen Beschaltungselemente auf die Ständerseite umgerechnet. Im ESB erkennt man sie an der Kennzeichnung durch Hochkommata.

[2]Seite 170
[3]Seiten 1-6
[4]Seite 26

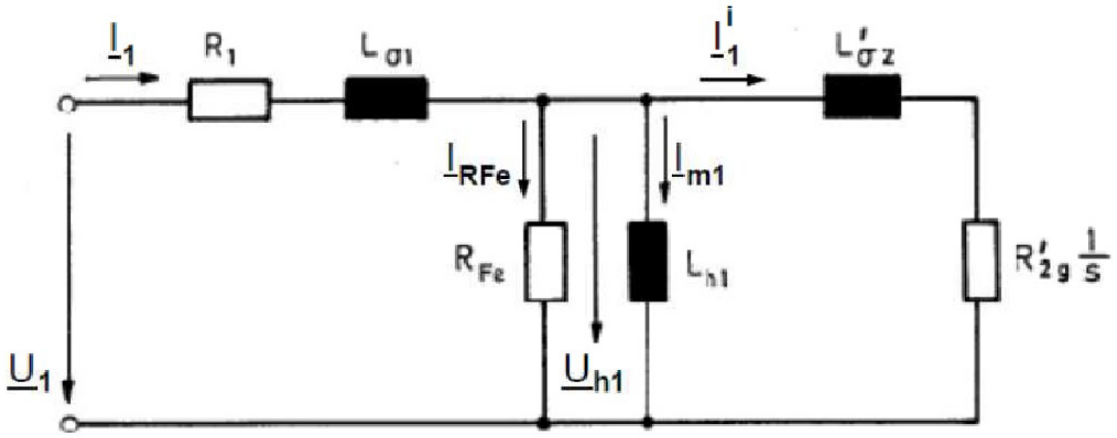

Abbildung 2.1: Ständerseitiges ESB der Asynchronmaschine

Das ESB berücksichtigt die Eisenverluste durch den Eisenverlustwiderstand R_{Fe}, die Streuinduktivitäten $L_{\sigma 1}$ und $L_{\sigma 2}{}'$, die Hauptinduktivität L_{h1}, den Ohmwiderstand R_1 der Ständerwicklung sowie den Widerstand $\frac{R_{2g}{}'}{s}$, der die Verluste im Rotor sowie die mechanische Belastung der Welle beschreibt.

Eine wichtige Größe zur Beschreibung einer Asynchronmaschine stellt der Schlupf s dar, der den relativen Unterschied zwischen Drehfeld und Motordrehzahl beschreibt [2][5].

Er lässt sich mit Hilfe der Gleichung

$$s = \frac{n_0 - n}{n_0} \tag{2.1}$$

berechnen.

Hierfür wird die synchrone Drehzahl benötigt, die sich wie folgt berechnet, wobei p die konstruktiv festgelegte Polpaarzahl ist:

$$n_0 = \frac{f_1}{p} \tag{2.2}$$

2.2 Bestimmung der Parameter des Ersatzschaltbildes

Da für die Benutzung des Programmes neben den Nenndaten der ASM auch die Parameter des ESBs benötigt werden, müssen diese zunächst messtechnisch bestimmt werden. Dazu werden die Strangwiderstände gemessen und ein Leerlauf- und

[5]Seite 172

ein Kurzschlussversuch durchgeführt. Eine Anleitung hierfür befindet sich auch in der Laborbeschreibung GEP 4 (s. Anhang A1). Zur konkreten Durchführung der Versuche siehe auch Kapitel 3.

Zu berücksichtigen ist zunächst, dass bei den Versuchen anstatt der Induktivitäten nach Abbildung 2.1 die Reaktanzen gemessen werden.

Weiterhin kann mit dem Kurzschlussversuch nur der Gesamt-Streublindwiderstand gemessen werden, für den gilt:

$$X_K = X_{\sigma 1} + X_{\sigma 2}{}' \tag{2.3}$$

Da die beiden Reaktanzen in der Realität annähernd gleich groß sind, werden diese vereinfachend gleichgesetzt.

$$\frac{X_K}{2} = X_{\sigma 1} = X_{\sigma 2}{}' \tag{2.4}$$

Mit diesen beiden Modifikationen ergibt sich nun das Ersatzschaltbild in Abbildung 2.2, mit dessen Hilfe das Verhalten von Asynchronmaschinen im folgenden untersucht wird.

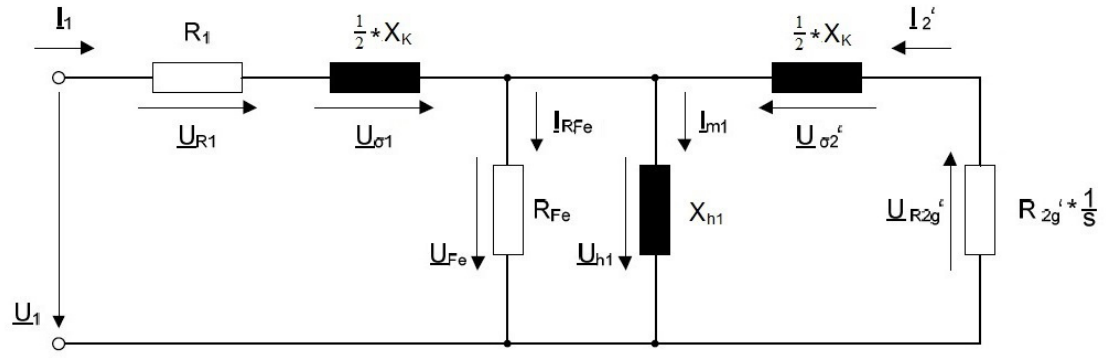

Abbildung 2.2: Modifiziertes ständerseitiges ESB

Der Ohmwiderstand der Ständerwicklung R_1 wird im Ruhezustand der ASM ermittelt. Da Widerstände temperaturabhängig sind, muss R_1 mit der Gleichung [1][6]

$$R_\vartheta = R_{20^\circ}(1+\alpha * \triangle T) \tag{2.5}$$

[6]Kapitel 3 Seite 5

und dem Temperaturbeiwert für Kupfer bei 20°C [1][7]

$$\alpha_{20^\circ} = 0.00393 \, \frac{1}{K} \tag{2.6}$$

auf die Betriebstemperatur der ASM umgerechnet werden.

Durch den Leerlaufversuch ($n \approx n_0$, $s \approx 0$) lassen sich der Eisenverlustwiderstand R_{Fe} und der auf die Ständerwicklung bezogene Hauptblindwiderstand X_{h1} mit den folgenden Gleichungen berechnen [3][8]:

$$R_{Fe} = \frac{3 * U_{1N}^2}{P_{FeN}} \tag{2.7}$$

$$I_{10WN} = \frac{P_{0N}}{3 * U_{1N}} \tag{2.8}$$

$$I_{10BN} = \sqrt{I_{10N}^2 - I_{10WN}^2} \tag{2.9}$$

$$X_{h1} = \frac{U_{1N}}{I_{10BN}} \tag{2.10}$$

Durch den Kurzschlussversuch ($n = 0$, $s = 1$) lassen sich folgende Größen berechnen [3][9]:

$$R_K = \frac{P_{KN}}{3 * I_{1N}^2} \tag{2.11}$$

$$Z_K = \frac{U_{1K}}{I_{1N}} = \sqrt{R_K^2 + X_K^2} \tag{2.12}$$

$$X_K = \sqrt{Z_k^2 - R_k^2} \tag{2.13}$$

$$R_{2g}{}' = R_K - \frac{R_{1\triangle}}{3} \tag{2.14}$$

2.3 Das Zeigerdiagramm

Das erste Diagramm, das später im Programm simuliert werden soll, ist das Zeigerdiagramm. In diesem werden alle Ströme und Spannungen des einphasigen ständerseitigen ESB nach Abbildung 2.2 dargestellt. Die folgende Abbildung enthält ein Beispiel für ein Zeigerdiagramm [2][10].

[7] Kapitel 3 Seite 5
[8] Seite 28
[9] Seite 29
[10] Seite181

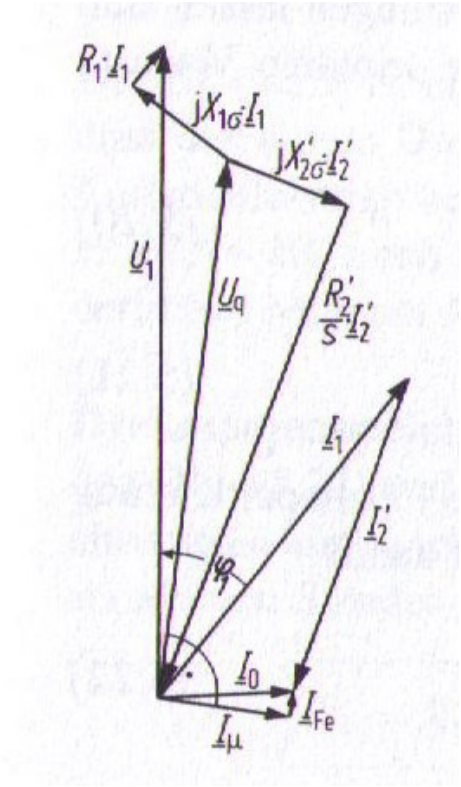

Abbildung 2.3: Beispiel für ein Zeigerdiagramm

Zur Berechnung der einzelnen Spannungen und Ströme muss zunächst der Strom $\underline{I}_1$ bestimmt werden. Hierfür wird die Gesamtimpedanz $\underline{Z}_{ges}$ des ESBs benötigt, die sich mit der Gleichung 2.15 ermitteln lässt:

$$\underline{Z}_{ges} = \frac{1}{\frac{1}{R_{Fe}} + \frac{1}{j*X_{h1}} + \frac{1}{j*\frac{X_k}{2} + \frac{R_{2g}{}'}{s}}} + R_1 + j*\frac{X_k}{2} \tag{2.15}$$

Mit $\underline{Z}_{ges}$ folgt für $\underline{I}_1$:

$$\underline{I}_1 = \frac{\underline{U}_1}{\underline{Z}_{ges}} = \frac{\underline{U}_1}{\frac{1}{R_{Fe}} + \frac{1}{j*X_{h1}} + \frac{1}{j*\frac{X_k}{2} + \frac{R_{2g}{}'}{s}} + R_1 + j*\frac{X_k}{2}} \tag{2.16}$$

Nun folgt mit Hilfe des ohmschen Gesetzes:

$$\underline{U}_{R1} = R_1 * \underline{I}_1 \tag{2.17}$$

$$\underline{U}_{\sigma 1} = j*\frac{X_k}{2} * \underline{I}_1 \tag{2.18}$$

Mit Hilfe der Maschenregel ist es nun möglich die Spannung $\underline{U}_{h1}$ zu bestimmen.

$$\underline{U}_{h1} = \underline{U}_1 - \underline{U}_{R1} - \underline{U}_{\sigma 1} \tag{2.19}$$

Mit Hilfe dieser Spannungen und des ohmschen Gesetzes lassen sich nun die restlichen Ströme und Spannungen berechnen.

$$\underline{I}_{RFe} = \frac{\underline{U}_{h1}}{R_{Fe}} \tag{2.20}$$

$$\underline{I}_{m1} = \frac{\underline{U}_{h1}}{j*X_{h1}} \tag{2.21}$$

$$\underline{U}_{\sigma 2} = j*\frac{X_k}{2}*\underline{I}_2{}' \tag{2.22}$$

$$\underline{U}_{R2g}{}' = \frac{R_{2g}{}'}{s}*\underline{I}_2{}' \tag{2.23}$$

Der für die vorangegangenen beiden Gleichungen benötigt Strom $\underline{I}_2{}'$ lässt sich wie folgt über die Knotenregel bestimmen:

$$\underline{I}_2{}' = -\underline{I}_1 + \underline{I}_{RFe} + \underline{I}_{m1} \tag{2.24}$$

2.4 Das Kreisdiagramm

Bei dem Kreisdiagramm handelt es sich um eine Stromortskurve, die den Ständerstrom $\underline{I}_1$ in Abhängigkeit vom Schlupf s beschreibt. Um den Kreis händisch zu konstruieren kann man entweder die charakteristischen Punkte ($s = 0$, $s = 1$, $s = \infty$) bestimmen oder man berechnet den Radius und die Mittelpunktskoordinate [2][11]. Der daraus konstruierte Kreis besteht quasi aus den geometrischen Endpunkten aller Stromzeiger von $\underline{I}_1(s)$ in der Gaußschen Zahlenebene.

Da es später mit Hilfe des Rechners möglich ist, sehr viele Punkte zu berechnen, wird auf eine Konstruktion wie oben beschrieben verzichtet. Stattdessen wird $\underline{I}_1(s)$ in den Grenzen $-\infty \leq s \leq \infty$ variert, und so viele Punkte berechnet, dass ein vollständiger Kreis entsteht. Bei der Berechnung der Stromortskurve wird wieder von dem modifizierten ESB (s. Abb 2.2) ausgegangen und es ergibt sich mit Gleichung 2.16 für $\underline{I}_1(s)$:

$$\underline{I}_1(s) = \frac{\underline{U}_1}{\underline{Z}_{ges}} \tag{2.25}$$

[11] Seiten 188-191

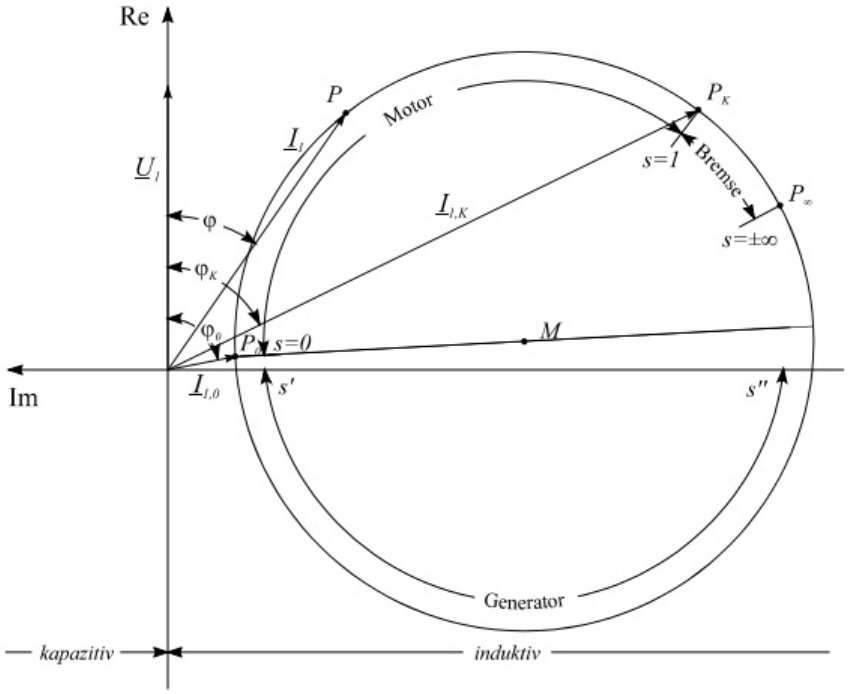

Abbildung 2.4: Beispiel für ein Kreisdiagramm

Wie in Abbildung 2.4 [6][12] zu sehen, kann man am Kreisdiagramm unter anderem die Betriebszustände einer ASM erkennen.

Für $0 < s < 1$ wird Wirkleistung aufgenommen und die ASM arbeitet als Motor. Wenn der Schlupf weiter erhöht wird, nimmt die Maschine neben elektrischer auch mechanische Wirkleistung auf und wirkt als Bremse. Wählt man $s < 0$ wird Wirkleistung abgegeben und die ASM ist im Generatorbereich.

2.5 Die Drehmoment-Drehzahl-Kennlinie

Die Drehmoment-Drehzahl-Kennlinie ordnet jeder Drehzahl ein Drehmoment zu. Anhand dieser Kennlinie kann man, wie schon beim Kreisdiagramm, die einzelnen Betriebszustände der ASM erkennen (s. Abb. 2.5) [6][13].

[12] Seite.12
[13] Seite 18

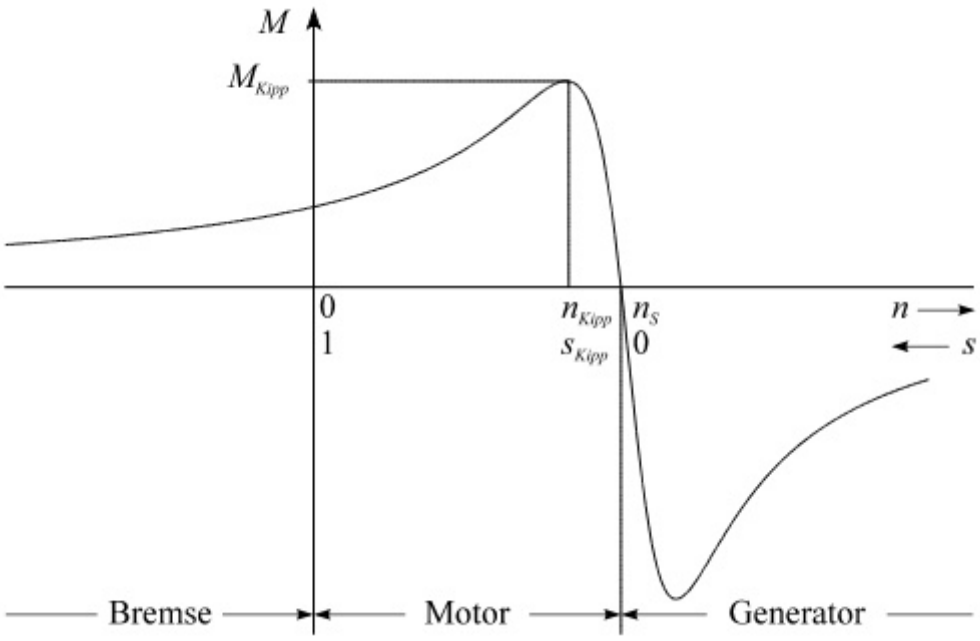

Abbildung 2.5: Beispiel einer Drehmoment-Drehzahl-Kennlinie

Zur Berechnung der Drehmoment-Drehzahl-Kennlinie wird das ESB (s. Abb. 2.2) erneut vereinfacht. Der Widerstand der Ständerwicklung und der Eisenverlustwiderstand werden vernachlässigt und es ergibt sich mit $R_1 = 0$ und $R_{Fe} = \infty$ folgendes neues ESB:

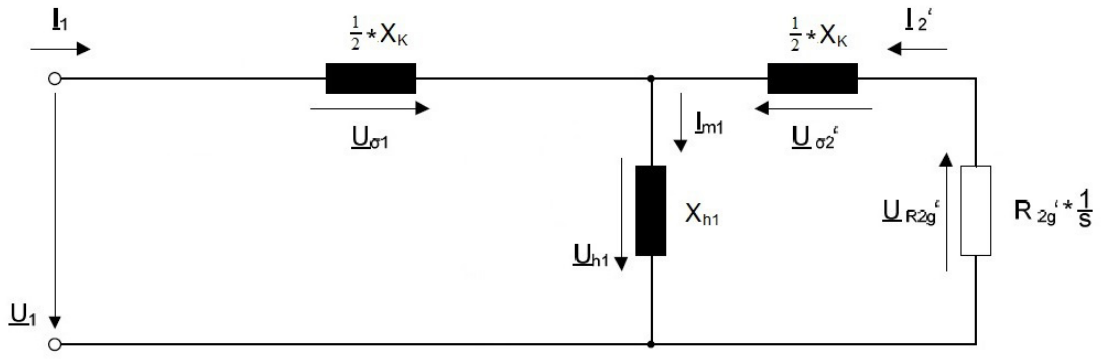

Abbildung 2.6: Vereinfachtes ESB

Zur Berechnung der Drehmoment-Drehzahl-Kennlinie wird zunächst die Streuziffer, der Kippschlupf und das Kippmoment benötigt, die sich wie folgt berechnen [3][14]:

Streuziffer

$$\sigma = 1 - \frac{L_{h1}^2}{(L_{h1}+L_{\sigma 1})*(L_{h1}+L_{\sigma 2}{}')} \tag{2.26}$$

Kippschlupf

$$s_k = \frac{R_{2g}{}'}{\omega_1 * \sigma * (L_{h1}+L_{\sigma 2}{}')} \tag{2.27}$$

Kippmoment

$$M_{ik} = \frac{3}{2} * \frac{1-\sigma}{\sigma} * \frac{p*U_1^2}{\omega_1^2*(L_{\sigma 1}+L_{h1})} \tag{2.28}$$

Da in dem Laborversuch, wie schon in Abschnitt 2.2 beschrieben, nicht die Induktivitäten sondern die Reaktanzen gemessen werden, müssen die Formeln für eine direkte Berechnung mit dem Programm zunächst mit folgendem Zusammenhang angepasst werden:

$$L = \frac{X}{\omega} \tag{2.29}$$

Da ω für alle Induktivitäten gleich ist, kürzt sich dies in der Formel heraus. Außerdem gilt auch hier die Annahme, dass Ständer- und Läuferreaktanz gleich groß sind und die Formel vereinfacht sich zu:

$$\sigma = 1 - \frac{X_{h1}^2}{(X_{h1}+\frac{X_k}{2})^2} \tag{2.30}$$

Für den Kippschlupf und das Kippmoment ergibt sich nun:

$$s_k = \frac{R_{2g}{}'}{\sigma * (X_{h1}+\frac{X_k}{2})} \tag{2.31}$$

$$M_{ik} = \frac{3}{2} * \frac{1-\sigma}{\sigma} * \frac{p*U_1^2}{\omega*(\frac{X_k}{2}+X_{h1})} \tag{2.32}$$

Die Grundlage für die Berechnung mit dem Programm ist die Kloss'sche Formel [3][15]:

$$\frac{M_i}{M_{ik}} = \frac{2}{\frac{s}{s_k}+\frac{s_k}{s}} \tag{2.33}$$

[14] Seiten 31-32
[15] Seiten 33-34

Für einen variablen Schlupf und damit für eine variable Drehzahl folgt:

$$M_i(s) = \frac{2*M_{Ik}}{\frac{s}{s_k}+\frac{s_k}{s}} \tag{2.34}$$

Wenn nun das Reibmoment Mr noch berücksichtigt wird ergibt sich [2][16]:

$$M(s) = M_i(s) - M_R = \frac{2*M_{Ik}}{\frac{s}{s_k}+\frac{s_k}{s}} - M_R \tag{2.35}$$

Hiermit lässt sich nun der Verlauf der Drehmoment-Drehzahl-Kennlinie berechnen.

[16] Seite 182

3 Messungen an einer realen Asynchronmaschine

Das während dieser Bachelorarbeit erstellte Programm soll, wie bereits in der Einleitung beschrieben, unter anderem zur Erweiterung des Laborversuchs GEP4 genutzt werden, welcher im 4. Semester an der HAW im Departement Informations- und Elektrotechnik durchgeführt wird. Die Aufgabenstellung zum Versuch befindet sich im Anhang in Anlage A1.

Da das erstellte Programm lediglich das stationäre Betriebsverhalten der Asynchronmaschine simuliert, wurden von dem Laborversuch GEP4 nur der Leerlaufversuch, der Kurzschlussversuch und die Messung der Strangwiderstände durchgeführt. Mit Hilfe dieser Messungen können alle für das einphasige ESB benötigten Werte bestimmt werden, so dass anschließend beispielhaft das Verhalten der Asynchronmaschine aus dem HAW Labor simuliert werden kann.

Die Messungen wurden am 12.06.2013 im Gebäude Berliner Tor 7 im Labor 03.01 am Versuchsstand 1 durchgeführt.

Die Nenndaten der ASM können dem Typenschild auf der ASM entnommen werden. Bei der ausgemessenen ASM sind dies folgende Werte:

$P_N = 4000\,W$

$n_N = 1440\,min^{-1}$

$\underline{U}_N = 400\,V$

$f_N = 50\,Hz$

$\underline{I}_N = 8.2\,A$

$cos\varphi_N = 0.83$

3.1 Messen der Strangwiderstände

Zur Bestimmung der Strangwiderstände wurde die Maschine zunächst von den Netzanschlüssen getrennt und anschließend die einzelnen Strangwiderstände gemessen. Zur Messung wurde das ABB Metrawatt M2036 verwendet. Die nachfolgende Tabelle enthält die Messergebnisse.

Strang	Widerstand [Ω]
U1	4.62
V1	4,65
W1	4,60

Tabelle 3.1: Messung der Strangwiderstände

Die Widerstände der einzelnen Stränge unterscheiden sich voneinander, darum wird für R_1 der arithmetische Mittelwert aller Widerstände gebildet.

Es ergibt sich:

$R_1 = 4,623\Omega$

Die Messung von R_1 fand bei Raumtemperatur statt, die circa 20°C betrug. Im Betrieb liegt die Temperatur der ASM jedoch wesentlich höher. Laut Aufgabenstellung für den Laborversuch wird von einer Bezugstemperatur von 75°C ausgegangen.

Mit der Gleichung 2.5 und dem Temperaturbeiwert nach Gleichung 2.6 folgt für den Widerstand bei Betriebsbedingungen:

$R_{1,75^\circ} = R_{1,20^\circ} * (1 + 0.00393\,\frac{1}{K} * (75^\circ\text{ -}20^\circ)$

$R_1 = 5,623\Omega$

3.2 Leerlaufversuch

Zur Bestimmung der Parameter R_{Fe} und X_{h1} wird der Leerlaufversuch durchgeführt.

Hierfür wird die ASM zunächst auf Betriebstemperatur gebracht und mit einem Temperaturfühler überwacht. Die Spannung an der ASM kann mit Hilfe eines Drehstromtransformators stufenlos eingestellt werden.

Zur Messung der Spannungen, der Ströme sowie der Wirkleistung wurde das Messgerät Hioki 3165 eingesetzt.

Die ASM wird während des Versuchs im Motorbetrieb gefahren. Die Spannung soll hierbei soweit vermindert werden, bis der Schlupf maximal 20% vom Nennschlupf beträgt.

Der Nennschlupf ergibt sich nach Gleichung 2.1 zu:

$s_N = \frac{n_0 - n_N}{n_0} = \frac{1500\,min^{-1} - 1440\,min^{-1}}{1500\,min^{-1}} = 0.04$

20% vom Nennschlupf ergeben sich dann zu:

$$s_{20\%} = s_N * 0.2 = 0.008 \tag{3.1}$$

Die dazugehörige minimale Drehzahl berechnet sich wie folgt:

$$n_{min} = n_0 * (1 - s_{20\%}) = 1488\, min^{-1} \tag{3.2}$$

Die Spannung U_1 wurde während des Versuchs ausgehend von 400 V in 20 V Schritten so lange vermindert, bis n_{min} erreicht wurde. Die nachfolgende Tabelle enthält die Messergebnisse.

U_1 in V	U_1^2 in V^2	I_{10} in A	P_0 in kW	n in min^{-1}
400	160000	4.72	0.376	1499
380	144400	4.06	0.334	1498
360	129600	3.51	0.296	1498
340	115600	3.15	0.270	1498
320	102400	2.87	0.251	1497
300	90000	2.58	0.232	1497
280	78400	2.39	0.220	1496
260	67600	2.20	0.208	1496
240	57600	2.00	0.194	1495
220	48400	1.85	0.184	1495
200	40000	1.68	0.174	1494
180	32400	1.54	0.165	1493
160	25600	1.40	0.156	1491
140	19600	1.30	0.149	1488
133	17689	1.28	0.140	1488

Tabelle 3.2: Messergebnisse des Leerlaufversuchs

Wenn man die Leistung über der Spannung zum Quadrat plottet, erhält man das folgende Diagramm.

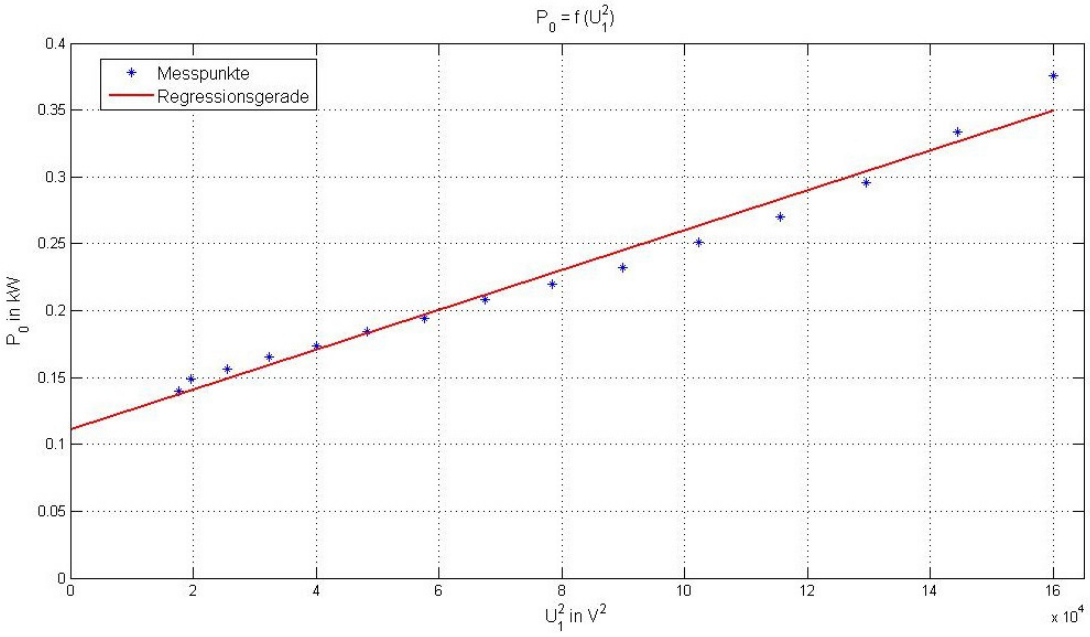

Abbildung 3.1: Messpunkte und Regressionsgerade

Da die Messpunkte keinen linearen Verlauf darstellen, wird dies mit Hilfe einer Regressionsgeraden angenähert (s. Abb. 3.1). Die Regressionsgerade lässt sich dabei durch die allgemeine Gleichung $y = 0.1112 + 0.000014890 * x$ beschreiben.

Der Schnittpunkt mit der Y-Achse ist eine konstante Verlustleistung, die durch Reibung hervorgerufen wird. Sie ergibt sich zu:

$$P_R = 0.1112kW$$

Hieraus lässt sich nun das Reibmoment M_R berechnen.

$$M_R = \frac{P_R}{2\pi n_0} = 0.71Nm \tag{3.3}$$

Zur Bestimmung des Widerstandes R_1 wird die Eisenverlustleistung P_{Fe} benötigt. Diese ergibt sich zu

$$P_{Fe} = P_{ges,N} - P_R = 0.34944kW - 0.1112kW = 0.23824kW \tag{3.4}$$

Damit folgt für Gleichung 2.7:

$$R_{Fe} = \frac{U_{L1N}^2}{P_{Fe}} = \frac{(400\,V)^2}{0.23824\,W} = 671.59\Omega$$

Um den Hauptblindwiderstand X_{h1} zu bestimmen muss die Gleichung 2.10

$X_{h1} = \frac{U_{1N}}{I_{10BN}}$

mit Hilfe der Gleichungen 2.8 und 2.9 gelöst werden. Damit ergibt sich:

$X_{h1} = 48.81\Omega$

3.3 Kurzschlussversuch

Für den Kurzschlussversuch ($s = 1$) wird die ASM bei stillstehender Welle betrieben. Hierfür wurde das Schutzblech an der Welle entfernt und der Strom zunächst soweit veringert, bis der Motor nicht mehr von alleine anlaufen kann. Anschließend wird die Welle manuell festgehalten und der Strom so lange erhöht, bis $I_1 = 1.2 * I_{1N}$ beträgt.

Es ergibt sich also ein maximaler Kurzschlussstrom von

$I_1 = 1.2 * 8.2A = 9.83A$

Für die folgenden Messungen wurde wiederum das Hioki 3165 Messgerät benutzt. Die untenstehende Tabelle enthält die gemessenen Werte.

U_1 in V	I_1 in A	P_K in kW
58.0	4.3	0.169
60.4	4.5	0.186
65.8	5.0	0.236
70.0	5.5	0.275
74.8	6.0	0.328
80.0	6.5	0.395
85.0	7.0	0.464
88.2	7.5	0.515
93.6	8.0	0.596
95.5	8.2	0.631
98.4	8.5	0.683
103.4	9.0	0.778
110.4	9.84	0.928

Tabelle 3.3: Messreihe beim Kurzschlussversuch

Aus den Messwerten und den Gleichungen 2.11 bis 2.14 lassen sich die restlichen Parameter für das ESB bestimmen.

$R_K = \frac{P_{KN}}{3*I_{1N}^2} = 3.128\Omega$

Für $R_{2g}{}'$ folgt mit einer im Dreick geschalteten ASM:

$R_{2g}{}' = R_k - \frac{R_{1\triangle}}{3} = 3.128\Omega - 1.874\Omega = 1.254\Omega$

mit

$Z_k = \frac{U_{1k}}{I_{1N}} = \sqrt{R_k^2 + X_k^2} = 11.61\Omega$

folgt

$X_k = \sqrt{Z_k^2 - R_k^2} = 11.18\Omega$

4 Simulation

Die Simulationen wurden mit Hilfe der Software und Programmiersprache Matlab (Version 2012b) programmiert, welche von der Firma “The MathWorks, Inc.” ent wickelt und vertrieben wird.

Matlab wird vor allem für numerischen Berechnungen, zur Programmierung und für Visualisierungen eingesetzt. Die Anwendungsgebiete sind breit gefächert. So ist es z.B. möglich neben einfachen Berechnungen auch Signalverarbeitungen, steuer- und regelungstechnische Aufgaben oder Bild- und Videoverarbeitungen durchzuführen.

Um das während dieser Thesis entwickelte Programm auszuführen, muss Matlab auf dem Rechner installiert sein. Dabei ist zu beachten, dass das Programm mit der Matlab Version 2012b oder höher ausgeführt werden muss, da Matlab nicht immer abwärtskompatibel ist, und sonst unvorhersehbare Fehler auftreten können [5].

4.1 Matlab GUIDE

Matlabverfügt über das Tool GUIDE zur Programmierung von graphischen Benutzeroberflächen sogenannten GUIs (graphical user interface). Zur Erstellung einer neuen GUI gibt man im Command Window der Matlab Entwicklungsumgebung den Befehl “guide” ein. Nun kann man ein neues GUIDE-Projekt erstellen, welches als .fig abgespeichert wird. In der GUIDE-Programmieroberfläche kann man vorgefertigte Buttons, Textfelder, Fenster zur Ausgabe von Plots etc. platzieren. Abbildung 4.1 zeigt die Umgebung mit einigen Objekten beispielhaft.

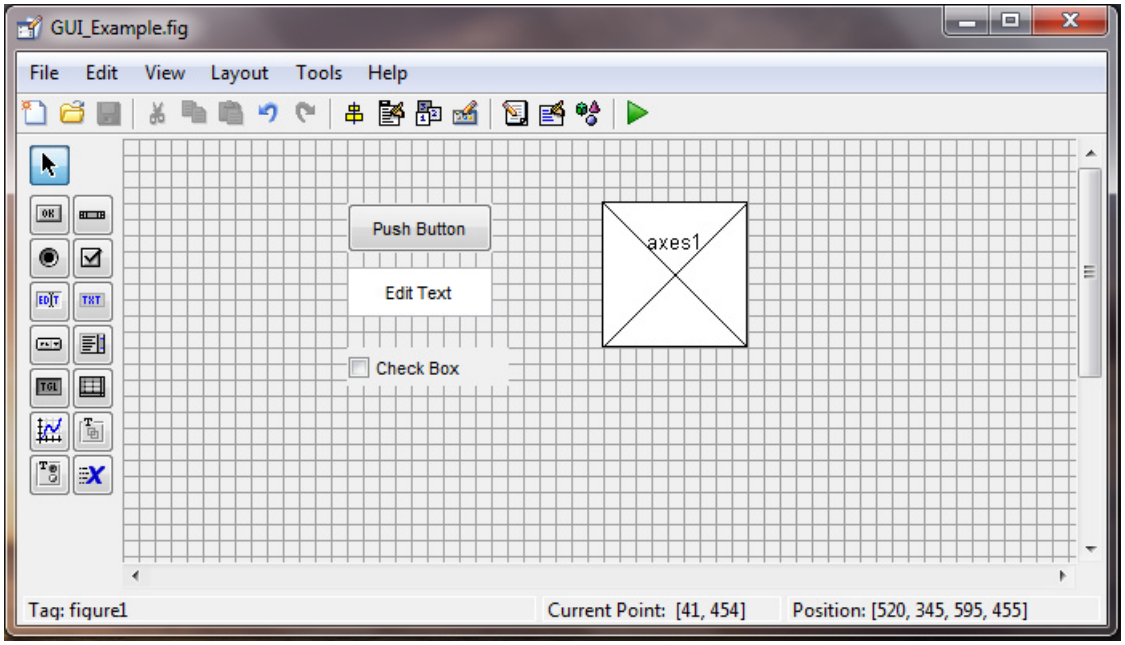

Abbildung 4.1: Beispiel GUIDE Entwicklungsumgebung

Beim Ausführen der Datei (durch Betätigung des grünen Buttons in der Symbolleiste oben rechts) wird automatisch ein dazugehöriges M-File erzeugt, in dem die grundlegenden Funktionen für alle Objekte hinterlegt sind. Diese Objekte sind jetzt noch "leere Hüllen", die ihre Funktionalität erst durch Programmierungen im M-File erhalten. In diesem müssen alle Berechnungen, Ausgabeanweisungen etc. programmiert werden.

4.2 Vorgehensweise bei der Programmierung

Da ich zu Beginn dieser Bachelorthesis lediglich Erfahrungen mit Matlab, aber nicht mit dem Tool GUIDE hatte, habe ich zunächst damit begonnen, mich mit Hilfe der Matlab Dokumentation und den dazugehörigen Tutorials in GUIDE einzuarbeiten. Dadurch konnte ich mir einen Überblick über die Programmierweise, aber auch über die Möglichkeiten und Grenzen des Tools verschaffen.

Nachdem ich mit der Programmierweise vertraut war, habe ich zunächst einen groben Entwurf der Benutzeroberfläche auf einem Blatt Papier entworfen. Dieser wurde während der Programmierphase immer wieder leicht verändert, diente jedoch als Orientierungshilfe beim Programmieren.

Nachdem die Oberfläche mit allen Objekten im GUIDE-Editor erstellt war, ging es anschließend um die Funktionalität. Hierbei bin ich so vorgegangen, dass ich kleine

Abschnitte, wie z.B. das Auslesen der Eingabefelder, im M-File programmiert habe und anschließend die Funktion umfangreich getestet habe. Dadurch konnten Fehler früh erkannt und behoben werden. Zusätzlich gibt es im Code zahlreiche Kommentare, die die Lesbarkeit fördern. All dies hat dazu geführt, dass eine aufwendige Fehlersuche am Ende vermieden wurde.

Im letzten Schritt folgte dann das designen der Benutzeroberfläche. Jetzt wurden alle Elemente im GUIDE-Editor so platziert, dass diese gleichmäßig ausgerichtet sind und von ihren Größen und Proportionen überein stimmen. Die Abbildung 4.2 zeigt die fertige GUIDE-Oberfläche mit allen 66 Objekten.

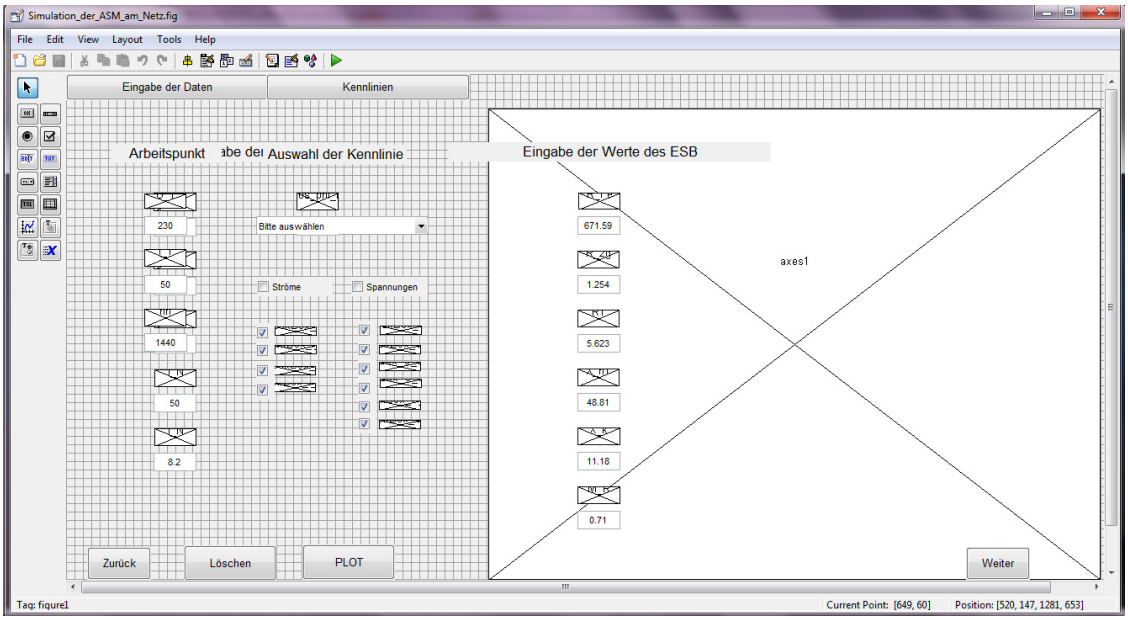

Abbildung 4.2: GUIDE-Oberfläche mit allen Objekten

4.3 Bedienungsanleitung für das Programm

Wie schon einleitend zu diesem Kapitel beschrieben, benötigt man zur Ausführung des Programms eine Matlab Version 2012b oder höher. Zunächst ist zu beachten, dass die beiden Dateien *Simulation_der_ASM_am_Netz.m* und *Simulation_der_ASM_am_Netz.fig* in demselben Ordner liegen. Zum Starten des Programms muss man die Datei *Simulation_der_ASM_am_Netz.m* mit Matlab öffnen. Das geöffnete M-File kann man nun durch die Funktionstaste F5 oder durch anklicken des grünen "Run"-Buttons ausführen. Anschließend öffnet sich die folgende Benutzeroberfläche:

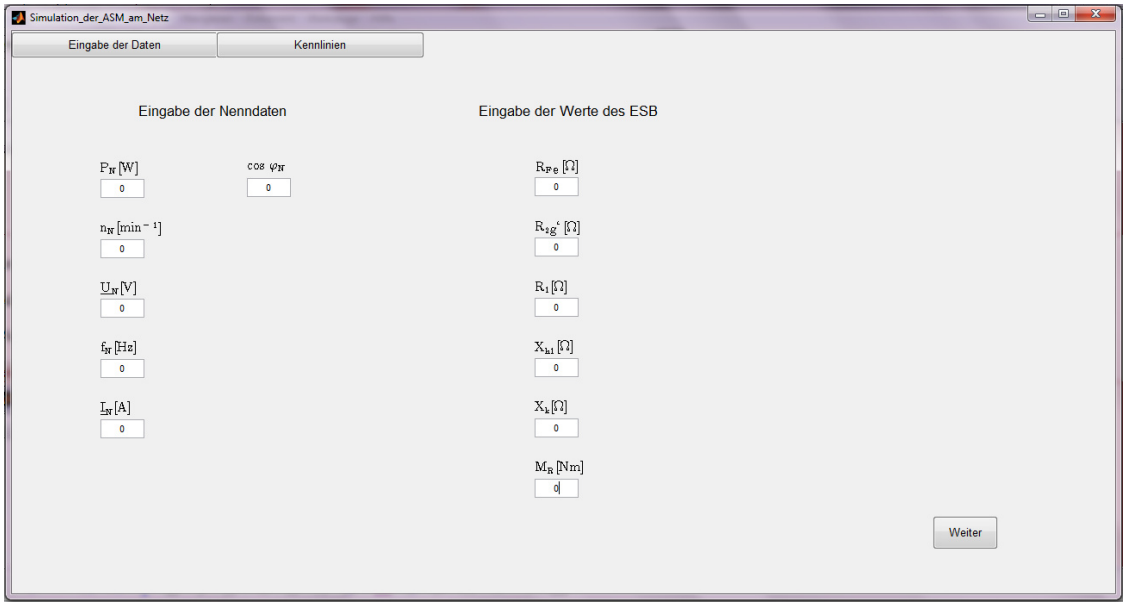

Abbildung 4.3: Benutzeroberfläche bei Programmstart

Wie in Abbildung 4.3 zu erkennen ist, gibt es Eingabefelder in denen die Nenndaten und die Parameter des ESB nach Abbildung 2.2 eingetragen werden sollen. Beim Programmstart enthalten alle diese Felder den Default-Wert null.

Nach Eingabe der Werte kann man entweder über den Button “Kennlinien” oder den Button “Weiter” zur nächsten Oberfläche gelangen, die in Abbildung 4.4 abgebildet ist.

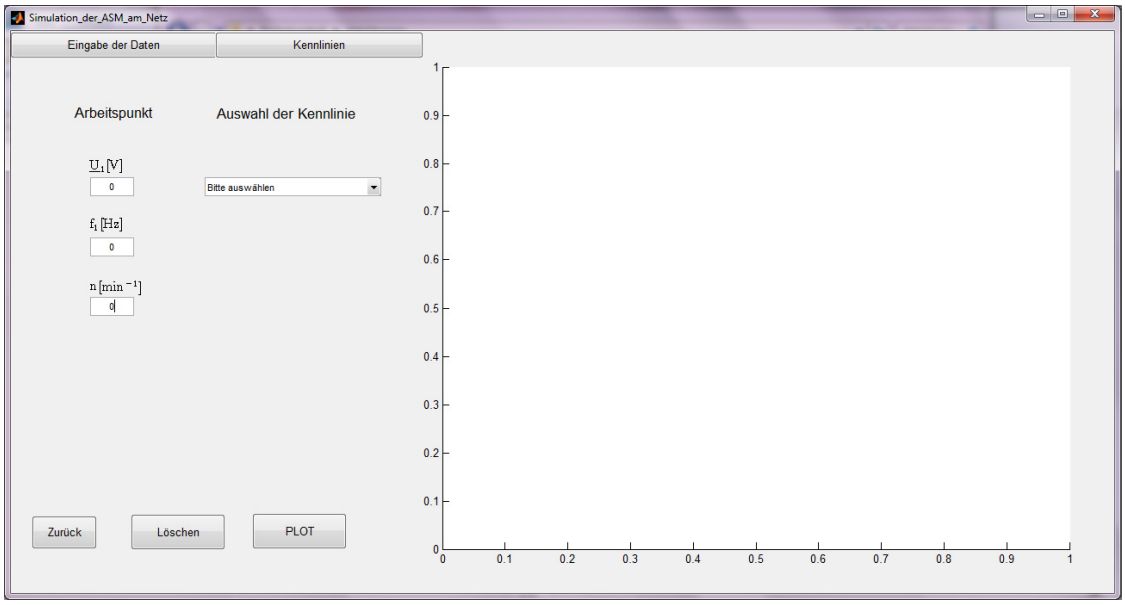

Abbildung 4.4: Zweite Benutzeroberfläche

Wie zu erkennen, kann man nun einen Arbeitspunkt für die Simulation in den Eingabefeldern vorgeben und in dem Popupmenü eine Kennlinie zur Simulation auswählen.

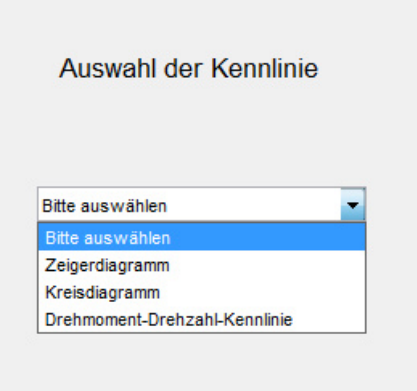

Abbildung 4.5: Popupmenü zur Auswahl der Kennlinie

Wenn man eine Kennlinie zur Simulation ausgewählt hat, kann man sich diese durch Betätigung des Buttons "PLOT" graphisch anzeigen lassen. Bei der Drehmoment-Drehzahl-Kennlinie und dem Kreisdiagramm besteht die Möglichkeit anschließend Parameter zu ändern und weitere Kennlinien zu plotten. Diese werden alle in demselben Diagramm angezeigt, was eine gute Vergleichbarkeit der Kennlinien untereinander ermöglicht. Löschen lässt sich die Graphik durch den "Löschen" Button.

Um zur Startseite zurückzugelangen um z.B. Eingabewerte zu ändern, kann man entweder oben den Button "Eingabe der Daten" oder unten den Button "Zurück" betätigen. Wenn man anschließend zur zweiten Oberfläche zurück wechselt, sind zuvor geplottete Graphiken wieder sichtbar, und man kann eine weitere Kennlinie hinzuplotten. Dies hat den Vorteil, dass es auch möglich, ist den Einfluss von Parametern des ESB's auf die Kennlinien zu untersuchen.

Eine Kurzanleitung zur Benutzung des Programms befindet sich auch in Anhang A2.

4.4 Spezielle Programmausschnitte

Für den Anwender wirkt das Programm später so, als wenn es zwei unterschiedliche Oberflächen gäbe (siehe Abschnitt 4.3). Dabei handelt es sich für den Programmierer eigentlich nur um eine Oberfläche (s. Abb. 4.2) auf der die einzelnen Elemente, je nach Zustand der Buttons und Checkboxen angezeigt oder ausgeblendet werden.

Der folgende Programmausschnitt zeigt einen Code-Ausschnitt beim Zurückschalten auf die erste Programmoberfläche. Mit dem dem Befehl *'Visible','Off'* wird ein Objekt unsichtbar, mit *'Visible','On'* wird ein Feld wieder sichtbar geschaltet.

```
694 -   set(handles.text8,'Visible','On');
695 -   set(handles.nextButton,'Visible','On');
696
697     %%%%%%%%%%%%%%%%%%%% OFF %%%%%%%%%%%%%%%%%%%%
698 -   set(handles.text16,'Visible','Off');
699 -   set(handles.text20,'Visible','Off');
700 -   set(handles.input_U1,'Visible','Off');
```

Abbildung 4.6: Programmauschnitt zum Ausblenden

Das Matlab Tool GUIDE stellt vorgefertigte Textfelder zur Verfügung, in die man einen String schreiben kann. Dies findet z.B. bei den Überschriften "Arbeitspunkt" oder "Eingabe der Kennlinien" Verwendung. Das Problem der Textfelder ist jedoch, dass diese keine Sonderzeichen wie z.B. Ω darstellen können, da keine Latex-Befehle eingegeben werden können. Um das Problem zu umgehen, muss man ein axes-Objekt erstellen, das eigentlich zur Ausgabe von Graphiken gedacht ist und dieses mit dem Befehl

set(handles.R_Fe,'Visible','Off');

unsichtbar machen. In ein axes-Objekt kann man nun wie in Abbildung 4.7 zu sehen mit Latex-Befehlen hineinschreiben. Dies ist ziemlich zeitaufwendig, da nicht alle Latex-Befehle unterstützt werden und die Positionen innerhalb des axes einzeln angegeben werden müssen.

```
122 -   text(0,0.2,'$\mathrm{R_F}$','interpreter','latex','FontSize', 12,...
123         'horiz','left','vert','middle','Visible','On','Parent',handles.R_Fe);
124 -   text(0.4,0.0,'$\mathrm{e}$','interpreter','latex','FontSize', 12,...
125         'horiz','left','vert','middle','Visible','On','Parent',handles.R_Fe);
126 -   text(0.6,0.2,'$\mathrm{[\Omega]}$','interpreter','latex','FontSize', 12,...
127         'horiz','left','vert','middle','Visible','On','Parent',handles.R_Fe);
```

Abbildung 4.7: Beispiel Latex als Text einfügen

Wie oben zu sehen, werden 6 Zeilen Code benötigt um "$R_{Fe}[\Omega]$" zu schreiben.

Alle in den folgenden Abschnitten 4.5 - 4.7 beschriebenen Berechnungen und Ausgaben werden in der Funktion

function plotButton_Callback(hObject, eventdata, handles),

die von Code-Zeile 947 bis zur Zeile 1307 geht, ausgeführt sobald der "PLOT"-Button betätigt wurde.

Zur Auswahl der richtigen Kennlinienberechnung in dieser Funktion gibt es eine Switch-Anweisung, welche die richtige Berechnung in Abhängigkeit von der im Popupmenü angewählten Kennlinie freischaltet.

Zu Beginn der Funktion *plotButton* werden die Variablen aus den Textfeldern eingelesen. Der Programmausschnitt in Abbildung 4.8 zeigt dies exemplarisch für die Parameter des ESB.

```
970     % read "ESB" values
971 -   R1 = str2double(get(handles.input_R1,'String'));
972 -   RFe = str2double(get(handles.input_RFe,'String'));
973 -   R_2g = str2double(get(handles.input_R2g,'String'));
974 -   Xh1 = str2double(get(handles.input_Xh1,'String'));
975 -   Xk = str2double(get(handles.input_Xk,'String'));
976 -   Mr=str2double(get(handles.input_Mr,'String'));
```

Abbildung 4.8: Einlesen der Parameter des ESB

Die Textfelder enthalten Strings, die zunächst mit der Funktion *str2double()* in eine Gleitkommazahl umgewandelt werden.

Eine weitere wichtige Berechnung ist die Anpassung der Reaktanzen auf ihren Widerstandswert bei der Frequenz im Arbeitspunkt. Abbildung 4.9 zeigt die Berechnung mit Matlab.

```
986     % calculate X  for other frequencys
987 -   Xh1=fa / f*Xh1;
988 -   Xk=fa / f*Xk;
```

Abbildung 4.9: Anpassung der Reaktanzen auf die Frequenz im Arbeitspunkt

4.5 Das Zeigerdiagramm

Wenn man das Zeigerdiagramm zur Simulation auswählt, erscheinen direkt danach zwei Checkboxen, mit denen man auswählen kann, ob man Spannungen und/oder Ströme simulieren möchte. Durch Aktivieren einer oder beider Checkboxen öffnen sich weitere Checkboxen, mit deren Hilfe man einzelne Spannungen bzw. Ströme auswählen kann (s. Abb. 4.10). Durch die Default-Einstellungen sind zunächst alle Spannungen/Ströme ausgewählt. Die Spannungen und/oder Ströme können anschließend geplottet werden.

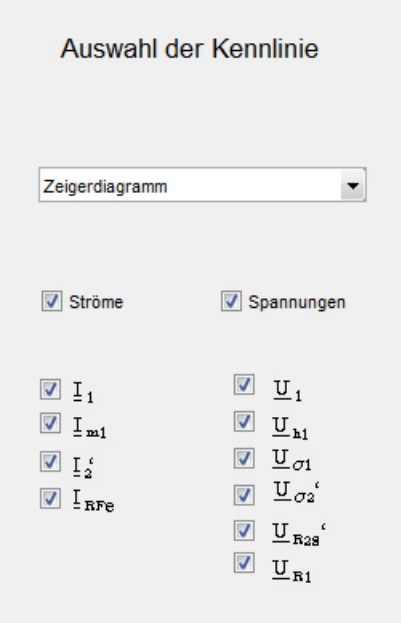

Abbildung 4.10: Checkboxen zur Auswahl von Spannungen und Strömen

Die Berechnung der Spannungen und Ströme erfolgt mit den Gleichungen 2.15 bis 2.24. Diese werden später vor der Ausgabe noch auf Nennstrom bzw. Nennspannung normiert.

```
1005        % calculate impedanz
1006 -      Z=1/(1/RFe+1/(1i*Xh1)+1/(1i*Xk/2+R_2g/sa))+R1+1i*Xk/2;
1007
1008        % calculate all voltages and currents
1009 -      I1=U1/Z;
1010 -      UR1=R1*I1;
1011 -      Usigma1=1i*Xk/2*I1;
1012 -      Uh1=U1-UR1-Usigma1;
1013 -      IRFe=Uh1/RFe;
1014 -      Im1=Uh1/(1i*Xh1);
1015 -      I2=-I1+IRFe+Im1;
1016 -      Usigma2=1i*Xk/2*I2;
1017 -      UR2g=R_2g/sa*I2;
```

Abbildung 4.11: Berechnung der Spannungen und Ströme für das Zeigerdiagramm

Da es bei dem Zeigerdiagramm möglich ist, sich beliebige Kombinationen von Spannungen und Strömen anzeigen zu lassen, musste ein Algorithmus entwickelt werden, um die Legende in der Graphik dynamisch so anzupassen, dass jeweils nur die angewählten Spannungen/Ströme in der Legende angezeigt werden. Hierfür wird zunächst die Anzahl der ausgewählten Checkboxen ermittelt. Dies passiert in den Zeilen 1030 bis 1071. Leider muss jede Checkbox einzeln mit eine If-Abfrage ausgewertet werden. Wenn eine Checkbox gesetzt ist, liefert die Zeile

get(handles.XXX, 'Value');

eine 1 zurück. Wenn die Anzahl an gesetzten Checkboxen bekannt ist, wird ein String-Array derselben Größe erstellt (s. Codezeile 1074). Dies ist notwendig, da das Array später nicht dynamisch erweitert werden kann.

Der folgende Programmausschnitt zeigt exemplarisch wie die Zeiger geplottet werden (s. Abb. 4.12).

```
1083      %I1
1084 -    startValue=0;
1085 -    arrow=I1/In;
1086 -    if get(handles.plot_I_1, 'Value')==1
1087 -       quiver(handles.axes1,real(startValue), imag(startValue), real(arrow), imag(arrow),'AutoScale','off','LineWidth',2);
1088 -       stringArray{counter}='I_1';
1089 -       counter = counter + 1;
1090 -    end
```

Abbildung 4.12: Beispiel zur Ausgabe eines Zeigers

In einem ersten Schritt werden der Startwert des Zeigers (Zeile 1084) sowie der Endwert (Zeile 1085) festgelgt. Hier findet gleichzeitig auch die Normierung statt. In der nächsten Zeile wird überprüft, ob die Checkbox, für in diesem Fall $\underline{I}_1$, überhaupt angewählt ist. Wenn dies so ist, wird zunächst der Zeiger geplottet (Zeile 1087). Dies geschieht mit dem Befehl *quiver( )* in dem man Start- und Endwert angibt. Zum Schluss wird der zugehörige String für die Legende in das Array geschrieben und die Zählvariable erhöht. Wenn alle angewählten Zeiger geplottet wurden, wird die Legende mit dem Befehl (Zeile 1188)

legend(handles.axes1,stringArray);

ausgegeben. Die Strings der Ströme und Spannungen sind in dem Array in der Reihenfolge abgelegt, in der die Pfeile geplottet wurden. Daduch findet nun beim Auslesen des Arrays die richtige Zuordnung zwischen Zeiger und der Legende statt.

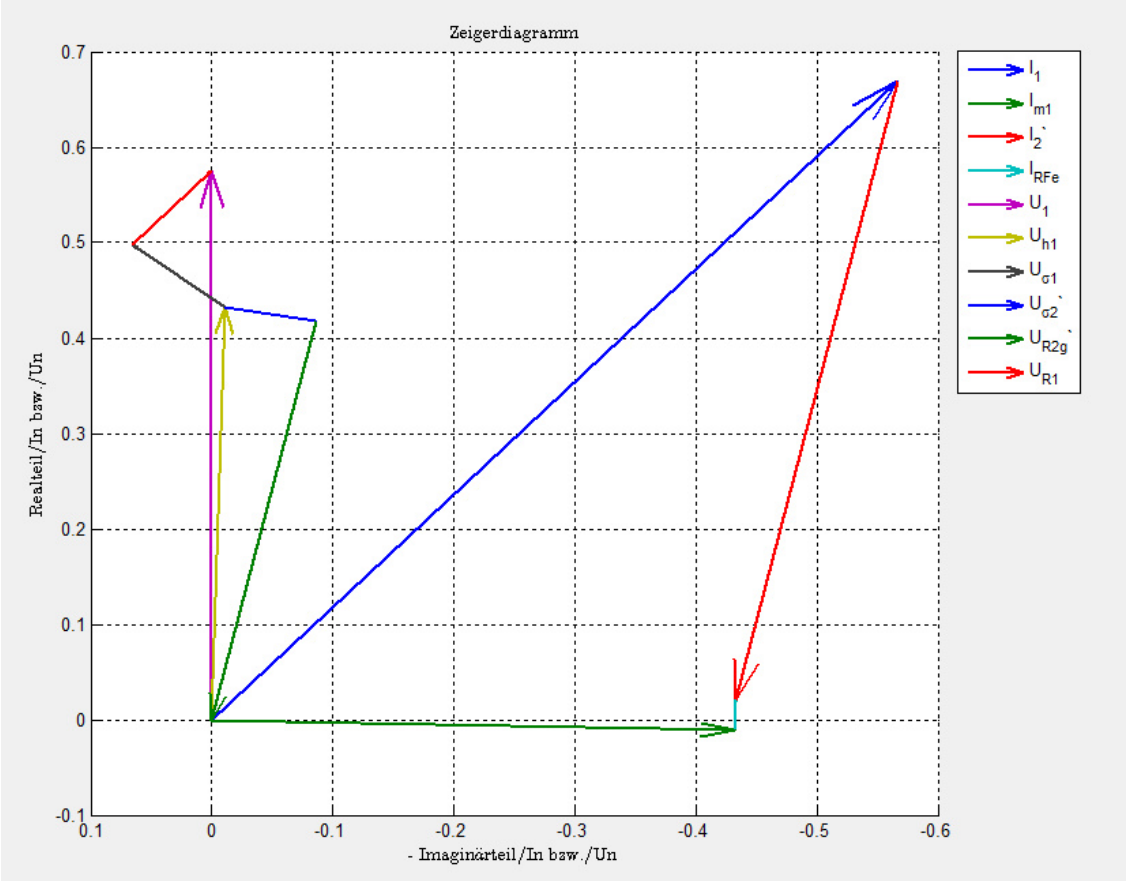

Abbildung 4.13: Simulation des Zeigerdiagramms

Abbildung 4.13 zeigt das vollständig simulierte Zeigerdiagramm mit den in Kapitel 3 ermittelten Daten des ESB.

Bei dem Zeigerdiagramm aber auch später bei dem Kreisdiagramm ist auf der X-Achse immer der negative Imaginärteil angegeben. Um die Ausgabe mit Matlab genau so hinzubekommen, muss das Diagramm vor der Ausgabe gedreht werden. Dies geschieht mit Hilfe folgender Zeile:

view(handles.axes1,270,90);

Die Darstellung der Pfeilspitzen ist abhängig von der Gesamtlänge des Pfeils, was dazu führt, dass z.B. der Pfeil für $\underline{I}_{RFe}$ in Abbildung 4.13 wie ein Strich wirkt. Wenn man sich allerdings nur diesen Pfeil anzeigen lässt, passt sich die ganze Graphik proportional an, und der Pfeil wird sichtbar (s. Abb. 4.14).

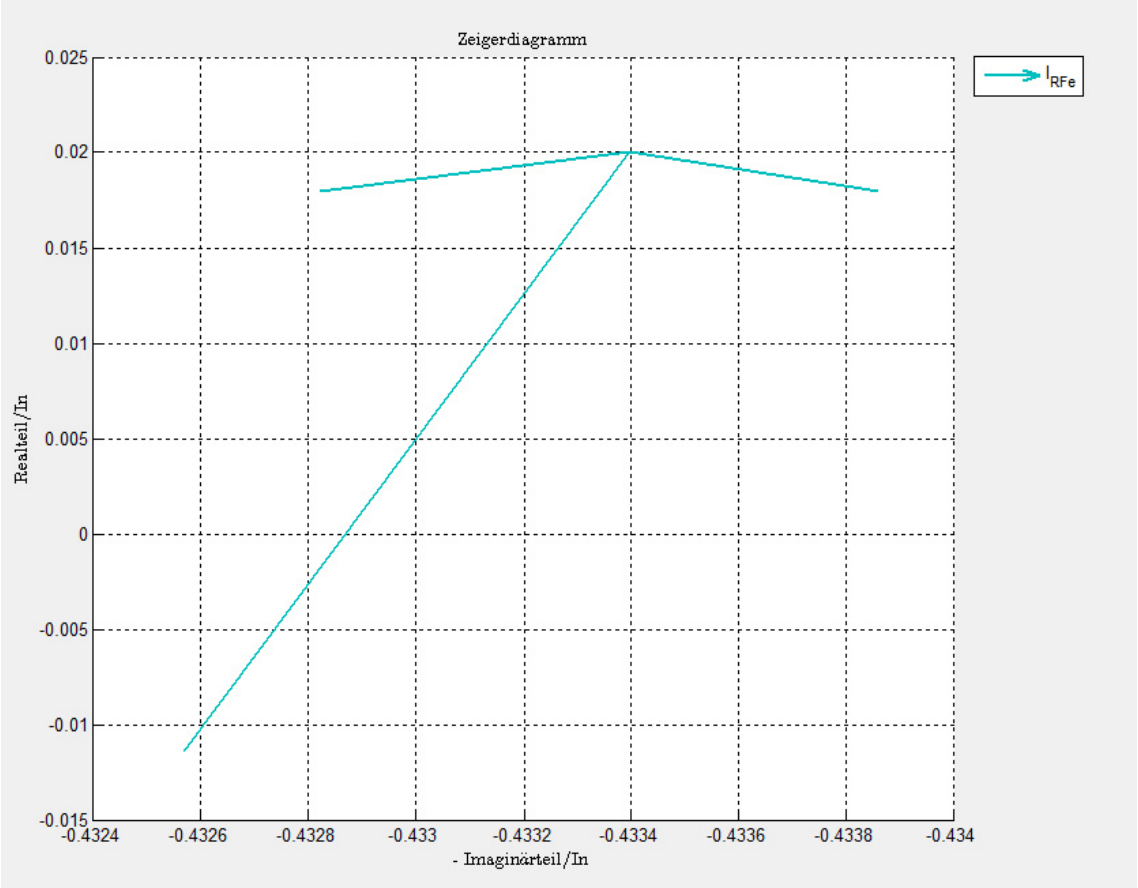

Abbildung 4.14: Simulation eines einzelnen Stromzeigers

4.6 Das Kreisdiagramm

Zur Berechnung der Stromortskurve wird wie in Abschnitt 2.4 beschrieben die Gleichung 2.25 verwendet und der Schlupf in den Grenzen $-\infty$ bis $+\infty$ variiert, so dass optisch ein Kreis aus diesen Punkten entsteht. Da die Kreisabschnitte in Abhängigkeit vom Schlupf nicht gleich verteilt sind, sondern z.B. der Kreisabschnitt zwischen $s = 1$ und $s = \infty$ wesentlich kleiner ist als der zwischen $s = 0$ und $s = 1$, werden die Punkte auf dem Kreis in logarithmischen Abständen berechnet.

Für s wird zunächst ein Vektor mit 100000 Punkten angelegt, die zwischen 10^{-10} und 10^{10} logarithmisch verteilt sind (s. Zeile 1200). Zusätzlich werden in den Zeilen 1201 und 1202 zwei weitere Vektoren der Größe s angelegt. Anschließend wird der Strom $\underline{I}_1(s)$ einmal für positiven und einmal für negativen Schlupf berechnet und die Ergebnisse in die beiden zuvor erstellten Vektoren geschrieben. Letztendlich besteht der Kreis also aus insgesamt 200000 Punkten besteht. Bei der Berechnung findet auch direkt eine Normierung auf $\underline{I}_N$ statt.

```
1199          % create logarithmic vector for current
1200 -        s=logspace(-10,10,100000);
1201 -        Ipos=zeros(size(s));
1202 -        Ineg=zeros(size(s));
1203
1204          %Calculate data
1205 -        for x=1:100000
1206 -            Ipos(x)=U1/(R1+1i*Xk/2+1/(1/RFe+1/(1i*Xh1)+1/(1i*Xk/2+R_2g/s(x))))/In;
1207 -            Ineg(x)=U1/(R1+1i*Xk/2+1/(1/RFe+1/(1i*Xh1)+1/(1i*Xk/2+R_2g/-s(x))))/In;
1208 -        end
```

Abbildung 4.15: Programmausschnitt zur Berechnung des Kreisdiagramms

Die Ausgabe des Kreises erfolgt mit dem plot-Befehl. Die Punkte, die in den beiden Vektoren gespeichert sind, werden hierbei nach Realteil zu Imaginärteil also konkret *real(Ipos)* zu *imag(Ipos)* bzw. *real(Ineg)* zu *imag(Ineg)* ausgegeben. Der Kreis besteht also eigentlich aus zwei zusammengesetzten Halbkreisen. Die anderen Parameter setzen die Farbe und die Strichstärke fest, mit der die Ausgabe erfolgen soll.

```
1220          % plot circle
1221 -        plot(handles.axes1,real(Ipos),imag(Ipos),'black', real(Ineg),imag(Ineg),'black','LineWidth',2);
```

Abbildung 4.16: Plotten des Kreises

Bei der Ausgabe des Kreises trat zunächst das Problem auf, dass dieser nicht rund war, sondern eher eiförmig. Dies lag daran, dass die Achsen nicht den gleichen Maßstab hatten. Dies kann jedoch mit der Code-Zeile

axis (handles.axes1, 'equal');

erzwungen werden, so dass ein Kreis entsteht.

Dies führte anschließend zu dem Problem, dass auch die anderen Kennlinien mit gleichem Achsenmaßstab dargestellt wurden, was unerwünscht ist. Deshalb steht in dem Code für das Zeigerdiagramm in Zeile 1023 und bei der Drehmoment-Drehzahl-Kennlinie in Zeile 1270 der Befehl

axis (handles.axes1, 'normal');

um wieder eine automatische Skalierung zu erzeugen.

Wie in Abbildung 4.18 zu sehen, werden neben dem Kreis auch spezielle Punkte eingezeichnet sowie ein Zeiger für den Nennpunkt geplottet. Die Ausgabe des Zeigers erfolgt wie die Ausgabe der Pfeile im Abschnitt Zeigerdiagramm beschrieben mit Hilfe der Funktion *quiver( )*. Die anderen Punkte wurden wie in Abbildung 4.17 beispielhaft für $s = \infty$ zu sehen, zunächst mit Gleichung 2.16 und ihrem speziellen Wert für s berechnet (Zeile 1249) und dann als roter Punkt in der Graphik eingezeichnet (Zeile 1250). Anschließend wird noch der Name des Punktes als String

ausgegeben. Hierfür wird der Endpunkt des zugehörigen Stromzeigers genommen und der Startpunkt für den auszugebenen String um -0.5 auf der imaginären Achse verschoben. Dadurch steht der Text später im Diagramm immer rechts neben dem roten Punkt.

```
1248        % plot s=infinity
1249 -      Is_inf=U1/(R1+1i*Xk/2+1/(1/RFe+1/(1i*Xh1)+1/(1i*Xk/2)))/In;
1250 -      plot(handles.axes1,real(Is_inf), imag(Is_inf),'x','LineWidth',3,'Color','r');
1251 -      text(real(Is_inf),imag(Is_inf)-0.5/In,'s = \infty','Parent', handles.axes1);
```

Abbildung 4.17: Beispiel für Ausgabe von speziellen Punkte

Mit den Nenndaten und den Parametern des ESB nach Kapitel 3 lässt sich nun die ASM aus dem HAW-Labor simulieren. Die folgende Abbildung zeigt das Ergebnis:

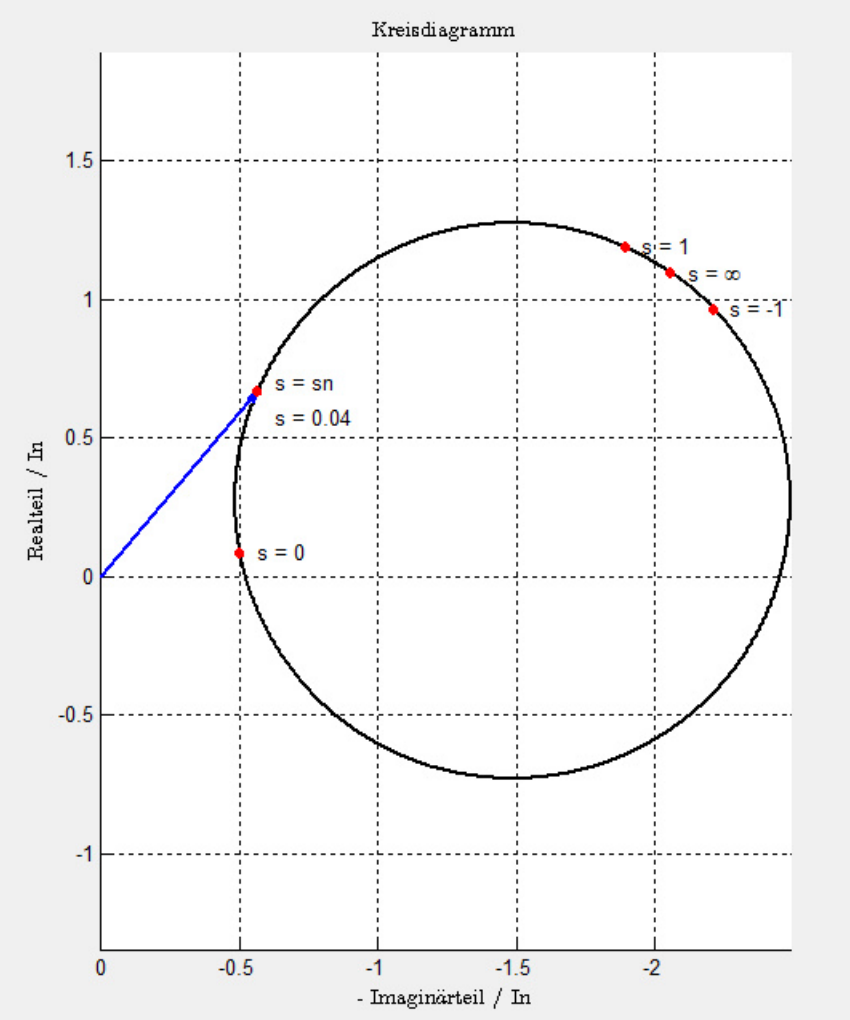

Abbildung 4.18: Kreisdiagramm mit den Daten der ASM aus dem HAW-Labor

Weiterhin ist es möglich, einzelne Parameter zu ändern und die Abhängigkeit der Stromortskurve von diesen zu untersuchen.

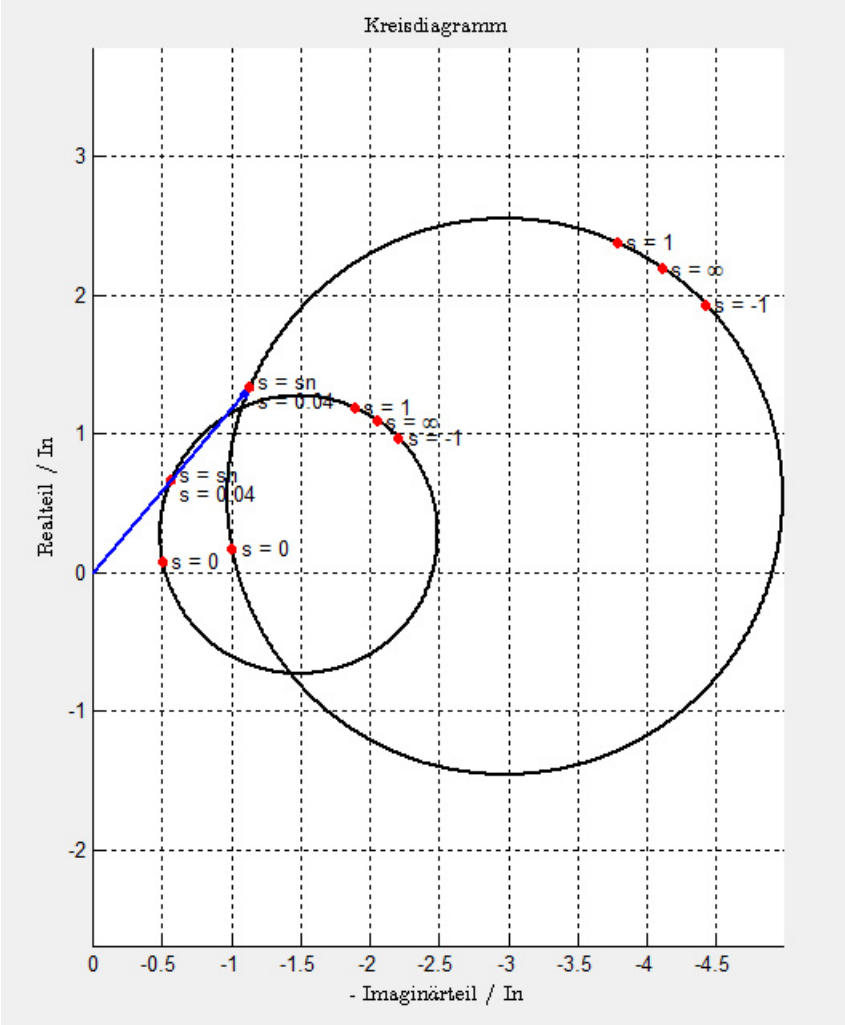

Abbildung 4.19: Kreisdiagramm bei unterschiedlicher Ständerspannung

Obige Abbildung zeigt das Kreisdiagramm einmal mit $\underline{U}_1 = 230\,V$ und einmal mit $460\,V$. Es zeigt sich, dass das Kreisdiagramm proportional abhängig von der Eingangsspannung $\underline{U}_1$ ist. Genauso kann die Abhängigkeit von anderen Parametern untersucht werden.

4.7 Die Drehmoment-Drehzahl-Kennlinie

Die Drehmoment-Drehzahl-Kennlinie wird wie in Abschnitt 2.5 beschrieben mit Hilfe des vereinfachten ESB (s. Abb. 2.6) und mit Hilfe der Kloss'schen Formel berechnet.

Hierfür werden zunächst die Streuziffer σ, der Kippschlupf s_k und das Kippmoment M_{ik} mit den Gleichungen 2.26 - 2.28 berechnet.

```
1281          % calculation
1282 -        sigma=1-(Xh1^2)/(Xh1+0.5*Xk)^2;
1283 -        sk=R_2g/(sigma*(Xh1+0.5*Xk));
1284 -        Mik=3/2*(1-sigma)/sigma*p*U1^2/(2*pi*fa*(0.5*Xk+Xh1));
```

Abbildung 4.20: Berechnung von σ, s_k und M_{ik}

Anschließend wird ein Vektor für Drehzahlen von $n = 0 - 3000\,min^{-1}$ angelegt, sowie ein Vektor derselben Größe für das zugehörige Drehmoment erstellt, welcher mit Nullen initialisiert wird. Anschließend wird noch die Formel zur Berechnung des Schlupfes im Programm bekannt gemacht.

```
1286          % create vector
1287 -        n=(0:1:3000);
1288 -        M=zeros (size(n));
1289 -        s=(n0-n)/n0;
```

Abbildung 4.21: Erstellen von Vektoren für n, M und Berechnung des Schlupfes

Jetzt wird in einer For-Schleife (Zeile 1292 - 1294), das jeweilige Drehmoment nach Gleichung 2.35 berechnet. Dieses wird auch direkt auf das Kippmoment M_{ik} normiert.

Weiterhin wird das Nennmoment berechnet (Zeile 1298) und anschließend die Kennlinie und der Nennpunkt ausgegeben. Das Nennmoment wird hierbei in Analogie zu den charackteristischen Punkten des Kreisdiagramms geplottet (Zeile 1303-1304).

```
1291          % calculate data
1292 -        for x=1:3001
1293 -            M(x)= ((2*Mik/(s(x)/sk+sk/s(x)))-Mr)/Mik;
1294 -        end
1295
1296          % calculate Mn
1297 -        sn=(n0-nn)/n0;
1298 -        Mn= ((2*Mik/(sn/sk+sk/sn))-Mr)/Mik;
1299
1300          % plot data
1301 -        plot(handles.axes1,n,M,'LineWidth',2)
1302          % plot Mn
1303 -        plot(handles.axes1,nn,Mn,'x','LineWidth',3,'Color','r')
1304 -        text(nn+50,Mn,'M_N','Parent', handles.axes1);
```

Abbildung 4.22: Berechnung des Drehmomentes und Ausgabe

Beim Testen des Algorithmus ist aufgefallen, dass es zu Fehlern kommt, wenn X_{h1} sehr groß gewählt wird. Ab einer bestimmten Größe kann Matlab die Berechnung nicht mehr ausführen, da diese numerisch durchgeführt wird und das Ergebnis ungültig wird. Um dies zu verhindern, wurde für die Drehmoment-Drehzahl-Kennlinie

ein oberer Maximalwert für X_{h1} von $10M\Omega$ festgesetzt. Dieser Wert hat sich bei Tests als hinreichen genau zur Simulation von $X_{h1} \rightarrow \infty$ erwiesen, da oberhalb dieses Wertes keine signifikanten Veränderungen auftreten. Außerdem können die Berechnungen numerisch stabil ausgeführt werden.

```
1276        %check Xh1 to avoid numeric problems
1277 -      if | Xh1 > 10000000
1278 -          Xh1 = 10000000;
1279 -      end
```

Abbildung 4.23: Begrenzung von X_{h1}

Wie schon zuvor beschrieben, kann nun die Abhängigkeit der Kennlinie von einzelnen Parametern untersucht werden. Abbildung 4.24 zeigt z.B. die Drehmoment-Drehzahl-Kennlinie der ASM aus dem HAW-Labor bei $f_1 = 25$, 50 und $75\,Hz$.

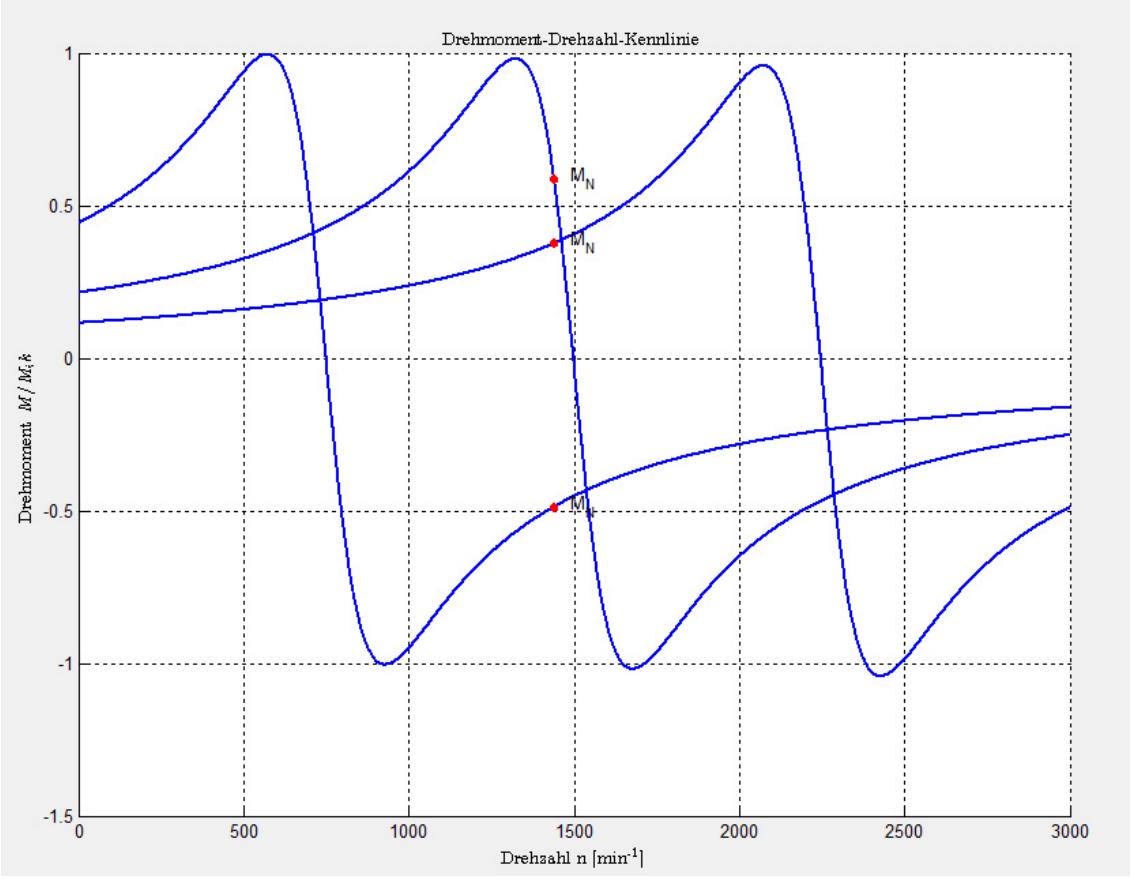

Abbildung 4.24: Drehmoment-Drehzahl-Kennlinie bei $f_1 = 25$, 50 und $75\,Hz$.

5 Fazit

5.1 Programmierung mit GUIDE

Die Programmierung mit Matlab GUIDE war etwas Neues für mich und hatte neben einigen Vorteilen auch Nachteile gegenüber anderen Programmierarten. Vorteilhaft war, dass auf viele vorgefertigter Bausteine (Buttons etc.) zurückgegriffen werden konnte, und diese lediglich um ihre Funktionalität ergänzt werden mussten. Ein weiterer großer Vorteil war, dass Diagramme relativ leicht erzeugt und dargestellt werden konnten.

Nachteilig war das es, wie schon in Abschnitt 4.4 beschrieben, keine Möglichkeit gibt Sonderzeichen und Latexbefehle in den Textfeldern zu benutzen. Deshalb musste man den sehr aufwendigen Weg gehen und die Zeichen einzeln in ein axes-Objekt schreiben. Diese Lösung war auf keinen Fall ideal sondern äußerst zeitaufwendig. Etwas einfacher war es, die Latex-Befehle in der Legende des Zeigerdiagramms einzufügen. Allerdings gibt es auch hier Einschränkungen im Befehlsumfang und es war z.B. leider nicht möglich komplexe Größen mit einem Unterstrich darzustellen.

Ein weiterer Mangel bei der Programmierung ist meiner Meinung nach, dass wie in Abbildung 4.2 zu sehen bei größerem Programmumfang die Oberfläche im GUIDE-Editor unübersichtlich wird und die Objekte übereinander liegen. Dies wird vor allem bei noch größeren Programmen schnell zu einem Problem.

Variablen sind nur innerhalb einer Funktion sichtbar und die Übergabe an andere Funktionen ist nicht so einfach möglich, was letztendlich dazu führt, dass auf globale Variablen zurückgegriffen werden muss. Dies sollte eigentlich bei einem guten Programmierstil vermieden werden.

Da Funktionen immer mit einem speziellen Objekt verbunden sind, kann man diese nicht häufiger aufrufen, was dazu führt, dass der Code häufig doppelt geschrieben werden muss. Dies betrifft z.B. die Funktionen für den Button “Kennlinien” und den Button “Weiter”, die eigentlich identisch sind, aber trotzdem beide den Code enthalten müssen, statt einer Funktion, die durch beide Buttons aufgerufen wird.

Alles in allem ist das Tool GUIDE gut geeignet, um kleinere Visualisierungen relativ schnell zu erstellen. Mit zunehmender Programmkomplexität wird es immer schwieriger sauber zu programmieren und es gibt immer wieder Einschränkungen, die mit Programmiersprachen wie Java, C# oder C++ besser zu lösen sind. Für das erstellte Programm war das Tool jedoch ausreichend.

5.2 Fazit und Ausblick zum Programm

Es ist gelungen, ein Programm zur Simulation des stationären Betriebsverhaltens von Asynchronmaschinen zu erstellen, das die Drehmoment-Drehzahl-Kennlinie, das Kreisdiagramm und das Zeigerdiagramm simulieren kann. Die Berechnungen beruhen auf dem einphasigen ständerseitigen Ersatzschaltbild der Asynchronmaschine. Lediglich für die Drehmoment-Drehzahl-Kennlinie wurde dieses vereinfacht. Dafür wird hier zusätzlich das Reibmoment berücksichtigt.

Leider ist es trotz zahlreicher Versuche nicht gelungen, die Graphiken nach der Simulation abspeichern zu können. Dies wäre sicherlich eine sinnvolle Erweiterung des Programms gewesen. Weiterhin gibt es auch Grenzen dessen was simuliert werden kann. So ist es z.B. nicht möglich die Sättigung von L_{h1} zu simulieren.

Der letzte Schritt wäre eigentlich gewesen, dass Programm im Studienalltag auf seine Praxisfähigkeit zu testen. Dies war zum Zeitpunkt der Fertigstellung dieser Bachelorarbeit nicht möglich, da während der Semesterferien weder Laborversuche noch Vorlesungen stattfinden. Trotzdem denke ich, dass sich das Programm bewähren wird. Es ist übersichtlich und intuitiv zu bedienen, was einen Einsatz zur Erweiterung des Laborversuchs GEP4 generell ermöglicht. Die Graphiken sind groß dargestellt und durch die angepasste Strichstärke kann man die Kennlinien auch aus größeren Entfernungen (z.B. im Hörsaal) gut erkennen. Die Genauigkeit der Simulationen ist mit den gemachten Vereinfachungen für die Berechnungen sicherlich ausreichend genau, um das stationäre Betriebsverhalten von Asynchronmaschinen im 4.Semester kennenzulernen und zu analysieren. All dies ermöglicht hoffentlich einen praktischen Einsatz während der Vorlesungen sowie im Labor.

Literaturverzeichnis

[1] Prof. Dr. Lipsmeier, Antonius (Hrsg.): Friedrich Tabellenbuch Elektrotechnik /Elektronik (2009), ISBN: 978-3-427-53025-1

[2] Rolf, Fischer: Elektrische Maschinen, Carl Hanser Verlag (2011), ISBN: 978-3-446-42554-5

[3] Prof. Dr. Röther, Michael: Grundlagen der Energietechnik Handout 04, Vorlesungsskript im Departement Informations- und Elektrotechnik der HAW-Hamburg (Wintersemester 2012)

[4] Ebinger, Swen: Simulation und Visualisierung des stationären Betriebsverhaltens der Asynchronmaschine. Bachelorthesis im Departement Informations- und Elektrotechnik der HAW-Hamburg (2012)

[5] http://www.mathworks.de/products/matlab/, [abgerufen am: 18.7.2013]

[6] http://antriebstechnik.fh-stralsund.de/1024x768/Dokumentenframe/Versuchsanleitungen/TU_Berlin/PR_ETiii_ASM.alt.pdf, [abgerufen am: 15.07.2013]

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

Anhang

Anhang A1: Versuchsbeschreibung zum Laborversuch GEP4

Anhang A2: Kurzanleitung zur Benutzung des Programms

Anhang A3: Matlab-Dateien

Die Anhänge A1 - A3 sind in elektronischer Form auf einer CD abgelegt und können bei Prof. Dr. Röther eingesehen werden.

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung §16(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, den 6. August 2013

_____
(*Tillmann Baer*)