Kurzeinführung in MATLAB / SIMULINK

Unterlagen zur Durchführung von Simulationsübungen
im Fachgebiet
Leistungselektronik und elektrische Antriebstechnik
Universität Paderborn, Mai 2004

(Vorlage freundlicherweise vom Fachgebiet Steuerungs- und Regelungstechnik zur Verfügung gestellt)

1 Kurzüberblick MATLAB

Matlab ist eine Softwareumgebung, die in erster Linie für numerische und ingenieurwissenschaftliche Anwendungen konzipiert ist. Matlab arbeitet auf der Basis von Matrizenrechnung und bietet zudem umfangreiche Möglichkeiten zur graphischen Darstellung von Ergebnissen. Nach dem Starten von Matlab wird ein Fenster, der sogenannte Arbeitsbereich (Bild 1.1), geöffnet. An der Kommandozeile („>>“) im Arbeitsbereich werden im allgemeinen die Befehle eingegeben und die Ergebnisse angezeigt.

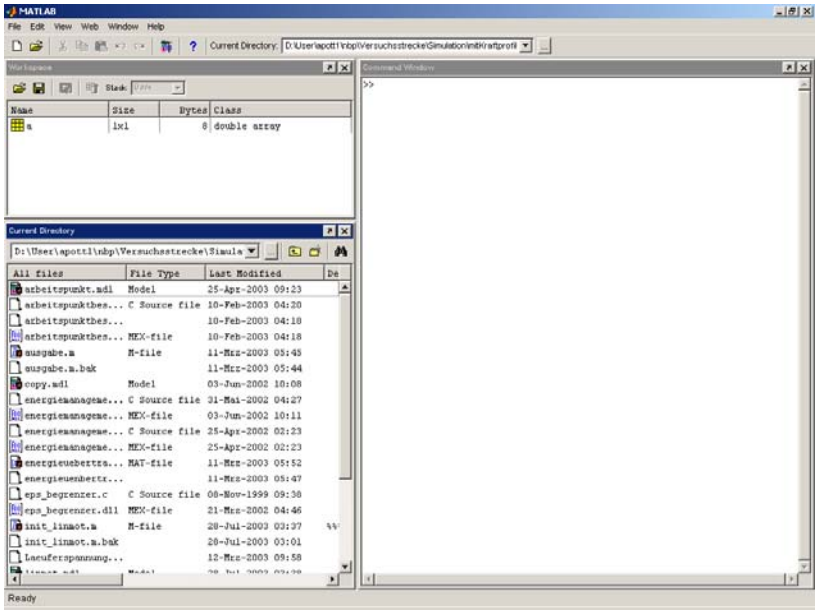

Bild 1.1 Matlab Arbeitsbereich

1.1 Zahlen, arithmetische Ausdrücke, Funktionen

Einige Beispiele für in Matlab verwendete **Zahlen**:

3 -99 0.0001

1.234E-20 5.678e9 -1.0e-2

2i -3.456i 7e8i

i steht für die komplexe Einheit $\sqrt{(-1)}$.

Arithmetische Operatoren: + , - , * , / , \ , ^ , '

Erläuterungen zu weiteren Operatoren mit:

>> `help !`

Funktionen: sin(x), cos(x), sqrt(x), sign(x), abs(x) ....

1.2 Zuweisungen und Variablen

Zuweisungen in Matlab haben im allgemeinen die Form

>> Variable = Ausdruck[;]

oder einfach nur

>> Ausdruck[;]

Wird dem Ausdruck keine Variable zugewiesen, erzeugt Matlab eine Variable mit Namen:

ans

Das Semikolon am Ende einer Eingabezeile unterdrückt die Ausgabe des Ergebnisses auf dem Bildschirm.

Für die Variablennamen gilt:

- Unterscheidung zwischen Groß- und Kleinschreibung
- Jeder Variablenname muß mit einem Buchstaben beginnen, anschließend können beliebige Zahlen oder Buchstaben folgen
- '.'und '_' sind ebenfalls im Variablennamen zugelassen;
- Matlab berücksichtigt nur die ersten 19 Zeichen eines Variablennamens;
- Alle Funktionsnamen sind klein geschrieben.

1.3 Vektoren und Matrizen

Matlab arbeitet im wesentlichen nur mit Vektoren und Matrizen. Die Vektor- bzw. Matrixelemente werden bei der Eingabe in eckige Klammern [ ] gefaßt.

Die Elemente eines Zeilenvektors werden durch Leerzeichen oder Kommata getrennt.

Beispiel 1: Zeilenvektor

```
>> z = [ 2 4 6]

z =

2 4 6
```

Die Elemente eines Spaltenvektors werden durch Semikolons oder Zeilentrennung (RETURN) getrennt.

Beispiel 2: Spaltenvektor

```
>> s = [ 1; 3; 5]

s =

1

3

5
```

Eine Matrix besteht aus Zeilen- u. Spaltenvektoren:

Beispiel 3: 3x3 Matrix

```
>> A = [ 1 2 3 ; 4 5 6 ; 7 8 9 ]

A =

1       2       3

4       5       6

7       8       9
```

Ein Matrixelement kann ein beliebiger Matlab Ausdruck sein. Der Ausdruck wird zum Zeitpunkt der Zuweisung ausgewertet.

Beispiel 4: Ausdrücke als Matrixelemente

```
>> x = [ -1.2 sqrt(3) (4+5+6)/7*8 ]

x =

-1.2000 1.7321 17.1429
```

Einzelne Matrixelemente können durch eingeklammerte Indizes angesprochen werden. Dabei wird die Länge der Matrix automatisch entsprechend den Indizes des neuen Elementes verlängert, wobei unbestimmte Zwischenelemente automatisch zu Null gesetzt werden.

Beispiel 5: Zugriff auf Matrixelemente

```
>> x(5) = abs(x(1))

x =

-1.2000 1.7321 17.1429 0 1.2000
```

Eine Matrix kann aus mehreren Matrizen zusammengesetzt werden.

Beispiel 6: Zusammengesetzte Matrizen

```
>> r = [ 10 11 12 ];

>> A = [ A ; r]

A =

1       2       3

4       5       6

7       8       9

10      11      12
```

1.4 Verwendung des Doppelpunktes

Der Doppelpunkt in der folgenden Anweisung erzeugt eine Zahlenfolge (Zeilenvektor) mit dem Startwert *begin*, der Schrittweite *width* und dem Endwert *end*:

begin [: *width*] : *end*

Wird die Schrittweite weggelassen, so wird sie auf 1 gesetzt.

Beispiel 7: Erzeugung eines Zeilenvektors mit Iterationswerten

```
>> c = 1 : 0.5 : 4

c =

1.0000 1.5000 2.0000 2.5000 3.0000 3.5000 4.0000
```

Beispiel 8: Untermatrizen

Untermatrizen einer Matrix können mit Hilfe des Doppelpunktes „:“ angesprochen werden.

```
>> B = A(2:3,:)

B =

4       5       6

7       8       9
```

2:3 erzeugt den Zeilenvektor *[2 3]*, der die zweite und dritten Zeile der Matrix A auswählt.

Der Spaltenindex wird zu : gesetzt. Der Doppelpunkt erzeugt den Zeilenvektor *[1 2 3]*, wobei die Dimension von A ausgewertet wird, d.h. der Start- bzw. Endwert wird durch den Kontext bestimmt.

1.5 Matrizen-Operationen

Matlab erlaubt die Addition, Subtraktion und Multiplikation von Matrizen unter Berücksichtigung ihrer Dimension.

Beispiel 9: Multiplikation einer Matrix mit einem Spaltenvektor

```
>> A = [ 1 2 3 ; 4 5 6 ; 7 8 9 ];
>> s = [ 1; 3; 5 ];
>> A*s
ans =
22
49
76
```

Ein Punkt vor einem Operator bedeutet, daß die Operation elementweise ausgeführt wird.

Beispiel 10: Elementweise Multiplikation von Vektoren

```
>> x = [1 2 3];
>> y = [4 5 6];
>> z = x.*y
z=
4 10 18
```

Beispiel 11: Elementweise Potenzierung

```
>> z = x.^y
z =
1          32          729
```

Beispiel 12: Vektoraddition

Will man einen Vektor a mit dem Einheitsvektor e_1 addieren, so müssen a und e_1 die gleiche Dimension besitzen.

```
>> a = [1 2 3];
>> e1 = [1 0 0];
>> a+e1
```

```
ans =
2     2     3
```

Ausnahme: Addition eines Vektors mit einem Skalar

```
>> a = [1 2 3];
>> s = [1]; % Achtung: s ist ein Skalar!
>> a+s
ans =
2     3     4
```

Beispiel 13: Transponieren einer Matrix

```
>> A = [ 1 2 3 ; 4 5 6 ; 7 8 9 ];
>> B = A'
B =
1     4     7
2     5     8
3     6     9
```

Beispiel 14: Inverse einer Matrix

```
>> C = [ 1 2; -0.5 1 ];
>> inv(C)
ans =
0.5000   -1.0000
0.2500    0.5000
```

Eine kleine Einführung in die Matrizen-Operationen erhält man auch mit dem Matlab Befehl *matmanip*.

1.6 Polynome

Matlab ermöglicht Berechnungen mit Polynomen, wobei die Koeffizienten als Zeilenvektoren behandelt werden.

Die Nullstellen des Polynoms $P(s) = a_n s^n + \ldots + a_2 s^2 + a_1 s^1 + a_0$ können mit der Funktion *roots(P)* berechnet werden. Ebenso ist es möglich aus den Nullstellen

N = [r1 r2 r3 ... rn]

des Polynoms *P(s)* die Polynomkoeffizienten a_i mit der Funktion *poly* zu berechnen.

Beispiel 15: Das Polynom

$P(s) = s^2 + 4s + 5$

wird durch den Zeilenvektor

```
>> P=[1 4 5];
```

eingegeben. Die Nullstellen werden durch

```
>> N=roots(P)
N =
-2.0000 + 1.0000i
-2.0000 - 1.0000i
```

bestimmt. Zur Kontrolle bilden wir die Polynomkoeffizienten

```
>> A=poly(N)
A =
1          4          5
```

Die Funktion *conv(a,b)* führt die Polynom-Multiplikation (*convolution*)

$$(a_n x^n + \ldots + a_1 x + a_0)(b_m x^m + \ldots + b_1 x + b_0)$$

durch.

Beispiel 16: Polynom-Multiplikation

```
>> a = [1 2 3 4];
>> b = [10 20 30];
>> c = conv(a,b)
c =
10         40         100        160        170        120
```

Der Vektor *c* enthält dann die Koeffizienten des Ergebnispolynoms, wobei die Dimension von *c* die Ordnung des Polynoms + 1 angibt:

```
>> ordnung = length(c)-1
ordnung =
5
```

Polynome spielen eine wichtige Rolle beim Umgang mit Übertragungsfunktionen von linearen, zeitinvarianten Systemen in MATLAB.

Beispiel 17: Bode Diagramm eines LZI-Systems mit der Übertragungsfunktion

$$G(s) = \frac{1 + 3s + 4s^2}{1 + 0,2s + 2s^2}$$

```
>> zaehler = [4 3 1]

zaehler =

4          3          1

>> nenner = [2 0.2 1]

nenner =

2          0.200      1
```

Der Befehl

```
>> bode(zaehler,nenner);
```

stellt dann Betrag und Phase des Bodediagramms im aktuellen Plot-Fenster dar.

Der Befehl

```
>> [betrag,phase,w]=bode(zaehler,nenner);
```

liefert Vektoren für Betrag, Phase und Frequenz in den links angegebenen Variablen zurück.

1.7 Graphische Darstellung von Wertefolgen

Matlab behandelt Wertefolgen ebenso wie Polynome als Vektoren. Neben umfangreichen Werkzeugen zur digitalen Signalverarbeitung bietet es die Möglichkeit der graphischen Darstellung.

Soll ein Signalverlauf in Abhängigkeit von der Zeit graphisch dargestellt werden, muß jeweils eine Wertefolge für das Signal *y* und eine Wertefolge für die Zeitbasis *x* vorliegen.

Beispiel 18: Graphische Darstellung von Wertefolgen

Zwei Signale mit den Wertefolgen *y1, y2* und der Zeitbasis *x* können durch den Befehl *plot(x, y1, x, y2)* jeweils über der Wertefolge der Variablen *x* dargestellt werden. Die Anzahl der Kurven in einem Diagramm ist beliebig und wird durch die Anzahl der paarweise übergebenen Wertefolgen festgelegt.

```
>> x = 0 : 0.1 : 2*pi;

>> y1= sin(x);

>> y2= cos(x);

>> plot(x, y1, x, y2);
```

Der Befehl *hold on* bewirkt, daß die folgenden Plots in das Koordinatensystem der zuletzt dargestellten Grafik gezeichnet wird. Mit *hold off* wird dieser Befehl wieder rückgängig gemacht.

Die folgende Liste enthält eine Auswahl an Matlab-Graphikfunktionen.

- *plot(X,Y)*
  Linearer 2-D Plot, der Vektor Y wird über dem Vektor X aufgetragen
- *loglog(X,Y)*
  wie plot, allerdings mit logarithmischer Skalierung
- *semilogx(X,Y)*
  wie plot, allerdings mit logarithmischer Skalierung der x-Achse
- *semilogy(X,Y)*
  wie plot, allerdings mit logarithmischer Skalierung der y-Achse
- *xlabel('text')*
  beschriftet die Abszissenachse mit dem String *text*
- *ylabel('text')*
  beschriftet die Ordinatenachse mit dem String *text*
- *title('text')*
  schreibt den Text im String *text* als Graphiktitel in den aktuellen Plot
- *grid*
  erzeugt ein Gitter im aktuellen Plot
- *hold on/off*
  verhindert/aktiviert das automatische Überschreiben der aktuellen Grafik

1.8 Hilfe-Funktionen

- *help*
  gibt eine Liste von Verzeichnissen aus, die verschiedene Matlab -Funktionen enthalten.
- *help* VERZEICHNISNAME
  gibt eine Liste mit Funktionsnamen und ihren Kurzbeschreibungen aus, die in dem entsprechenden Verzeichnis enthaltenen sind.
- *help* FUNKTIONSNAME
  gibt eine Beschreibung der gewählten Funktion auf dem Bildschirm aus.
- *helpdesk*
  öffnet die HTML Version des normalen help Befehls im Standardbrowser des Computers.
- *lookfor* NAME
  zeigt eine Auflistung aller Funktionen und Kommandos, die in ihrem Funktionsnamen oder in ihrer Kurzbeschreibung den Text NAME enthalten.
- *which* FUNNAME
  zeigt den volle Pfadname der Funktion FUNNAME an.

Zur Vertiefung der Kenntnisse sollten die Hilfe-Funktionen ausgiebig zum Nachschlagen der in dieser Anleitung erwähnten Matlab-Befehle verwendet werden.

1.9 Programmieren in MATLAB

Matlab erlaubt die Zusammenfassung mehrerer Anweisungen in sogenannten m-Files. Dies sind ASCII-Dateien mit der Dateinamenerweiterung „.m“, die über die Eingabe des Dateinamens (ohne Dateinamenerweiterung) an der Matlab Eingabeaufforderung aufgerufen werden können. Die in den m-Files enthaltenen Befehle werden dann der Reihe nach ausgeführt. Die m-Files können wiederum in anderen m-Files als Befehl auftauchen und so beliebig oft verschachtelt werden.

Die von anderen Programmiersprachen bekannten Kontrollstrukturen (FOR- bzw. WHILE-Schleifen, IF-THEN-Abfragen, BREAK) können in m-Files ebenfalls verwendet werden. Auf diesem Weg können z.B. komplette Abläufe in einem m-File *Skript* zusammengefaßt werden, was zu einer benutzerdefinierten Erweiterung des MATLAB-Befehlssatzes genutzt werden kann.

Neben den m-File *Skripts* können auch m-File *Funktionen* erstellt werden, die über einen eigenen Variablengültigkeitsbereich verfügen und denen über eine Parameterliste beliebige Funktionswerte übergeben werden können. Ergebnisse können dann über eine Liste von Variablen an die aufrufende Instanz zurückgegeben werden.

Matlab m-File *Skripts* und *Funktionen* können mit Hilfe eines beliebigen ASCII-Editors erstellt werden.

Beispiel 1: m-File *Skript* zur Darstellung von sin(t), cos(t)

Inhalt der ASCII-Datei „sincos1.m“

```
t=0:0.1:10;

y1=sin(t);

y2=cos(t);

plot(t,y1,t,y2);

grid on;

xlabel('t / s');

ylabel('blau: sin(t), grün: cos(t)');

title('Plot mit m-File Skript');
```

2 Kurzüberblick SIMULINK

Simulink ist eine Programmerweiterung von Matlab zur Simulation dynamischer Systeme. Zum Start wird *simulink* an der Kommandozeile im Arbeitsbereich von Matlab eingegeben.

Simulink verwendet Blockdiagramme zur Darstellung von dynamischen Systemen. Die einzelnen Blöcke können dabei aus verschiedenen Bibliotheken kopiert oder auch selbst erstellt werden.

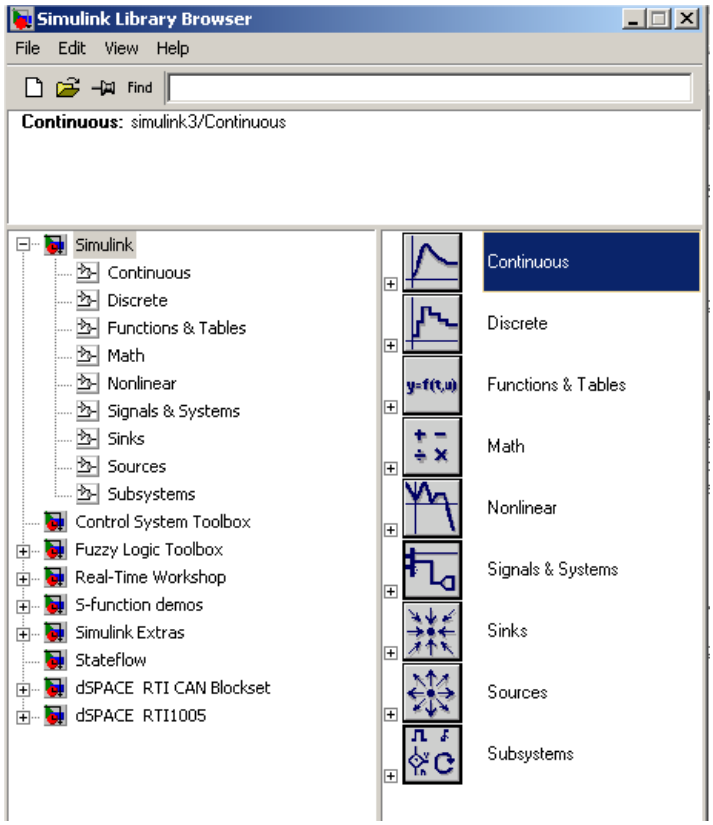

Bild 2.1 Blockbibliothek SIMULINK

Zur Eingabe eines neuen Modells wird zunächst im Menü FILE der Menüpunkt NEW gewählt. Daraufhin erscheint ein neues Arbeitsfenster, in dem das neue Modell erstellt werden kann.

Das Öffnen der im Simulink-Fenster angezeigten Bibliotheken erfolgt durch Doppelklick mit der linken Maustaste. Es erscheint daraufhin ein Unterfenster mit verschiedenen Blöcken, die durch einfaches Drag-and-Drop in das Arbeitsfenster kopiert werden können. Zum Kopieren kann auch das EDIT-Menü verwendet werden. Die Verbindung der einzelnen Blöcke erfolgt dadurch, daß man den Mauszei-

ger vom Ausgang des einen Blocks zum Eingang des nächsten bewegt, während gleichzeitig die linke Maustaste gedrückt wird. Das Anknüpfen an eine andere Verbindungslinie kann durch Drücken der rechten Maustaste erreicht werden.

Die Parameter der einzelnen Blöcke können über ein Dialogfenster eingestellt werden, welches sich nach Doppelklick mit der linken Maustaste auf den entsprechenden Block öffnet. Hier finden sich auch einige Erklärungen zu den einzelnen Blökken.

Es gibt verschiedene Möglichkeiten eine Simulation unter Simulink durchzuführen. Für den Anfang reicht die Bedienung der Simulation über die Menüleiste aus.

Die Simulationsparameter können nach Auswahl des Menüpunkts SIMULATION/ PARAMETERS verändert werden.

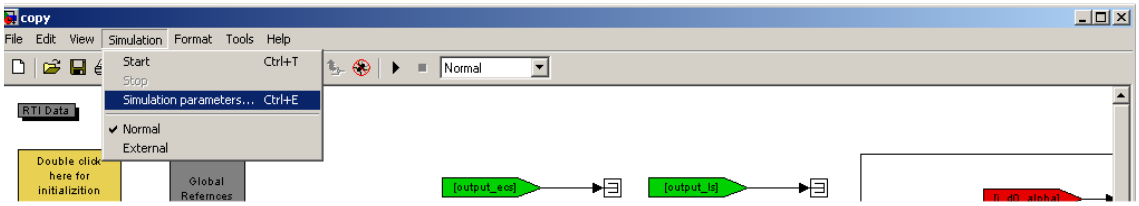

Bild 2.2 Menüpunkt „Simulation“

Simulink stellt mehrere Integrationsverfahren für die Simulation zur Verfügung. Aufgrund des unterschiedlichen Verhaltens der dynamischen Systeme, gibt es keine einheitliche Methode um jedes Modell genau und effizient zu simulieren. Die richtige Wahl des Integrationsverfahrens und der Simulationsparameter im Simulink-Menü (Bild 2.3) sind deshalb für die Simulation sehr wichtig und an das zu simulierende System anzupassen.

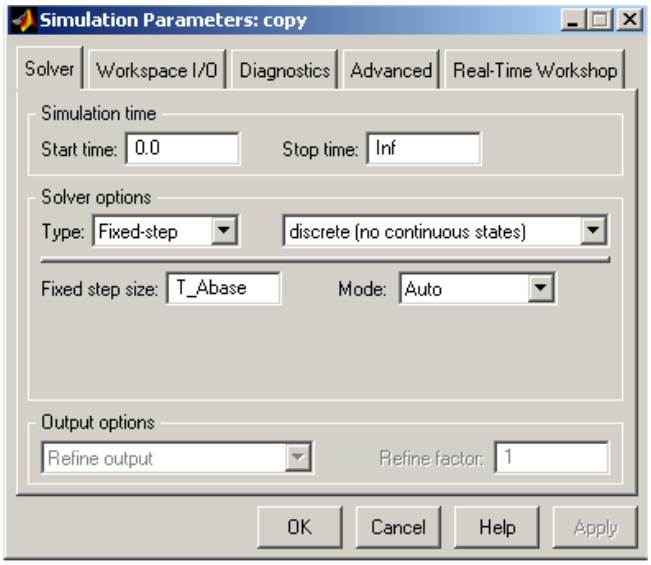

Bild 2.3 Simulationsparameter im SIMULINK Menu

Die Simulation wird durch Auswahl von SIMULATION/START oder durch Drücken des „play“ -buttons in der Symbolleiste (Bild 2.2) gestartet.

3 Aufgabe

Der für die Regelung des Magnetlagers notwendige Stromregler kann vereinfacht durch ein P-T1-Glied nachgebildet werden.

1. Erstellen Sie ein Simulationsmodell für diesen Stromregler, wobei der Strom auf 10A begrenzt werden soll. Verwenden Sie dazu die Parameter Ti für die Ersatzzeitkonstante und I_SMAX für den maximalen Strom. Zur Sollwerterzeugung des Stromes sollen die beiden Simulink-Blöcke *Step* und *Signal Generator* verwendet werden. Die Stromistgröße und Stromsollgröße sollen jeweils mit einem Block *Scope* aufgezeichnet werden und mit dem Block *To Workspace* sollen die zeitlichen Verläufe der beiden Größen in den Matlab-Arbeitsbereich übertragen werden.
   Parameter: $T_i = 99,6\mu s$,
   $I_{SMAX} = 10A$

2. Erstellen Sie ein m-File *init.m*, indem die Parameter des Simulationsmodells gesetzt werden und welches somit vor einer Simulation aufgerufen werden muss. Beginnen Sie das m-Skript mit dem Befehl *clear all*.

3. Führen Sie eine Simulation über eine Dauer von 10 ms durch, wobei die Simulationsparameter im Simulink Menue entsprechend einzustellen sind. Wählen Sie die Parameter im Block Step und Signal Generator so, dass der sich ergebende Sollstrom während der Simulation in die Begrenzung läuft.

4. Erzeugen Sie ein m-File *ausgabe.m* mit welchem die aufgezeichneten Stromgrößen und ihre Differenz grafisch dargestellt werden können. (Erzeugen Sie hierbei nur zwei *Figure* indem Sie auf Funktionen wie *subplo*t zurückgreifen. Die Achsen in den Darstellungen sind zu beschriften und mit dem Befehl *axis([ ])* ist der Darstellungsbereich zu optimieren.)

5. Stellen Sie den Frequenzgang des Stromreglers dar.