Elektrotechnik und Informationstechnik **Labor Regelungstechnik** **Prof. Dr. Hildenbrand**	**Kleine Anleitung zum Simulationsprogramm** **MATLAB SIMULINK**	Hochschule Offenburg University of Applied Sciences Letzte Änderung: 9. 5.2012 Birk

Was ist MATLAB?

MATLAB ist eine technische Programmiersprache für numerische Berechnungen und Darstellungen. **MATLAB** integriert numerische Analysis, Matrixberechnungen, Signalverarbeitung und Grafik in einer einfach benutzbaren Umgebung, in welcher Probleme und Lösungen ohne traditionelle Programmierung in einfacher mathematischer Beschreibung ausgedrückt werden können.

Der Name **MATLAB** steht für Matrix Laboratorium.
MATLAB ist ein interaktives System, in welches die Matrix das Grund-Datenelement ist. Es werden keine Dimensionierungen benötigt.

Was ist SIMULINK ?

SIMULINK ist eine Toolbox von **MATLAB** und gleichzeitig ein Programm zur Simulation von dynamischen Systemen. **SIMULINK** enthält viele spezielle Funktionen, die für eine solche Simulation nötig sind.

SIMULINK wird in der Regel auf zwei Arten genutzt:

- Erzeugung eines Modells **- Analyse eines Modells**

Eine typische Bearbeitung eines Modells mit SIMULINK beginnt mit der Analyse der Strecke bzw. des zu regelnden Systems. Daraus wird ein Modell erzeugt, der passende Regler dimensioniert. Anschließend kann das Modell simuliert und analysiert werden.

Einführendes Beispiel für den Umgang mit MATLAB SIMULINK:

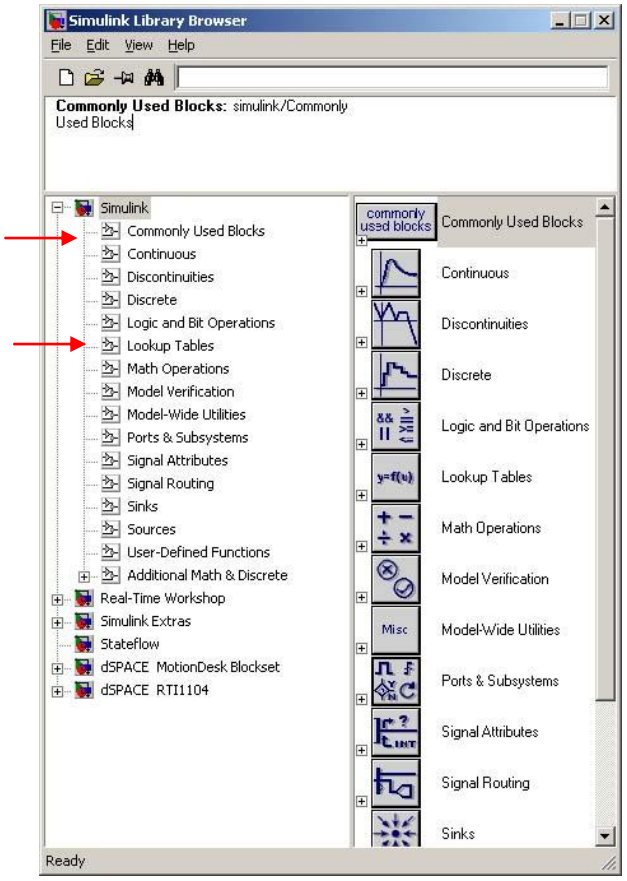
Um **MATLAB SIMULINK** anwenden zu können, muss man zuerst lernen, wie man Blöcke benutzt und Modelle erzeugt. Man muss sich außerdem mit den verfügbaren Arten von Blöcken vertraut machen. Letztendlich muss man auch lernen, wie man mit den Berechnungswerkzeugen umgehen kann. Im folgenden Abschnitt wird anhand eines Beispiels gezeigt, wie man ein einfaches Modell erstellt und danach simuliert.

1. Erstellung eines einfachen Modells:

SIMULINK verwendet die bildliche Darstellung von Blöcken (auch Icons genannt), um dynamische Modelle zu realisieren. Blöcke sind letztlich nichts anderes als Algorithmen. Durch das Zeichnen eines Blockdiagramms wird ein System erzeugt. Die verschiedenen Blöcke werden nicht selbst gezeichnet, sondern aus bestehenden Bibliotheken heraus kopiert, entweder aus den von **SIMULINK** mitgelieferten Standardbibliotheken oder aus selbst erzeugten Bibliotheken. Die Standardblockbibliothek ist unterteilt in verschiedene Untersysteme, gruppierte Blöcke entsprechend ihrer Eigenschaften. Blöcke können von diesen oder anderen Bibliotheken oder von bestehenden Modellen in Ihr Modell kopiert werden.

Um mit **SIMULINK** arbeiten zu können, muss erst **MATLAB** 

geladen werden.

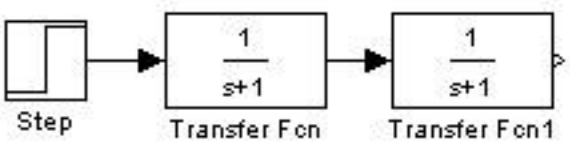


Im Matlab-Main-Window klickt man das unten gezeigte Icon für **SIMULINK** an.

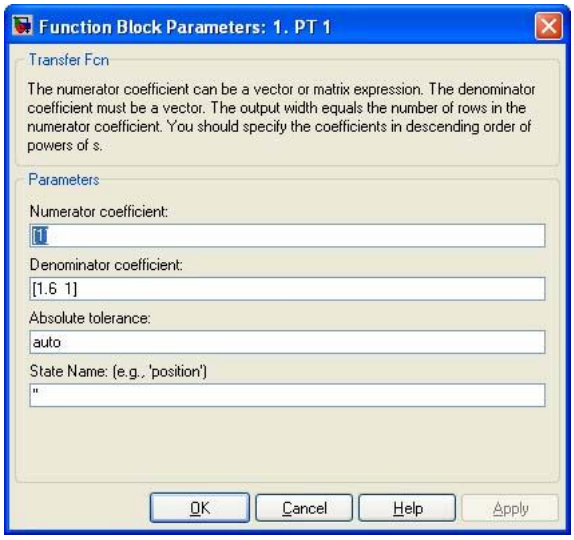
Das sich neu öffnende Kommandofenster "Simulink Library Browser" bietet zunächst die Möglichkeit, wie unter Windows üblich ein schon vorhandenes Modell zu laden oder ein neues zu erstellen. Darunter befindet sich eine Liste der verschiedenen Simulink-Block-Gruppen wie z. B. „Continuous“, „Sources“, „Sinks“ usw. Die „Sources“ Bibliothek enthält z. B. Blöcke, die in der Lage sind Signale zu erzeugen wie z. B. „Step“ zur Erzeugung einer Sprungfunktion oder „Clock“ als Zeitgeber.
Klicken wir eine dieser Gruppen an, erscheinen rechts davon die Blocksymbole dieser Gruppe. Bei „Sinks“ z. B. „Display“, „Scope“ oder „To Workspace“, welches wir benutzen. Einfacher ist es, ein **vorhandenes Modell** zu bearbeiten. Man öffnet dieses wie unter Windows üblich, in dem man in dem entsprechenden Verzeichnis auf den Dateinamen des Modells mit der Endung *.mdl klickt.
Die im Modell enthaltenen Blöcke können Sie wie folgt bearbeiten, kopieren, anders platzieren: Den zu bearbeitenden Block einmal anklicken und gewünschte Funktion ausführen.
Bei Doppelklick können Block-Parameter verändert werden. Werden weitere Blöcke des gleichen Typs benötigt, können diese einfach durch kopieren erzeugt werden. Auch mit den Cursor-Tasten können markierte Elemente schrittweise verschoben, manchmal auch präziser platziert werden. Man kann auch mehrere Blöcke durch Einrahmen mit der Maus gemeinsam markieren und bearbeiten.
Die Blöcke werden entsprechend des Signalflusses miteinander verbunden ("verdrahtet"). Man klickt mit der linken Maustaste auf den Ausgang des betreffenden Blocks, hält diese gedrückt und zieht eine "Leitung" zum Eingang des gewünschten Blockes. Bei Einzeleingängen kann man auch den sendenden Block markieren und danach mit gedrückter Strg.-Taste den Zielblock anklicken. Die Verdrahtung erfolgt dann automatisch.
Muss 1 Leitung an eine andere angeschlossen werden, Strg.-Taste drücken und gleichzeitig Mausklick (linke Taste) auf die Leitung, an der die neue Leitung angeschlossen werden soll. (Linke) Maustaste gedrückt halten und Leitung zum gewünschten Eingang ziehen.

Beispiel Regelstrecke mit 2 Verzögerungsgliedern :

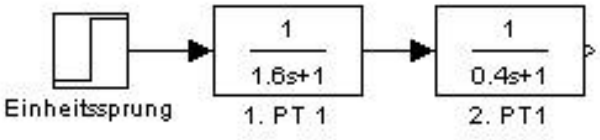
Es soll die Sprungantwort (Übergangsfunktion) einer Strecke, bestehend aus 2 PT1 - Gliedern, nach einem Führungsgrößen-Sprung aufgenommen werden.
Aus dem **Simulink-Library-Browser** erstellen wir ein neues, noch leeres Fenster für unser Modell. Beginnen wir mit der eigentlichen Regelstrecke. Von der Bibliothek klicken wir aus der Gruppe "**Continuous**" auf das Verzögerungsglied (Transfer Function) und ziehen es in das Fenster des neuen Modells. Wir kopieren diesen Block oder ziehen einen zweiten aus dem Library-Fenster.
Als Führungssprung verwenden wir die Step-Funktion, die wir in der Gruppe "**Sources**" finden. Auch diesen Block ziehen wir in das Modell-Fenster. Nun "verdrahten" wir die Ausgänge auf die folgenden Eingänge.



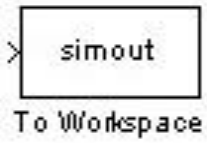
Rechts die Blöcke, wie wir sie aus der SIMULINK-Bibliothek erhalten.
In der Gruppe “**Commonly Used Blocks**“ befinden sich Blöcke, die häufiger benutzt wurden.



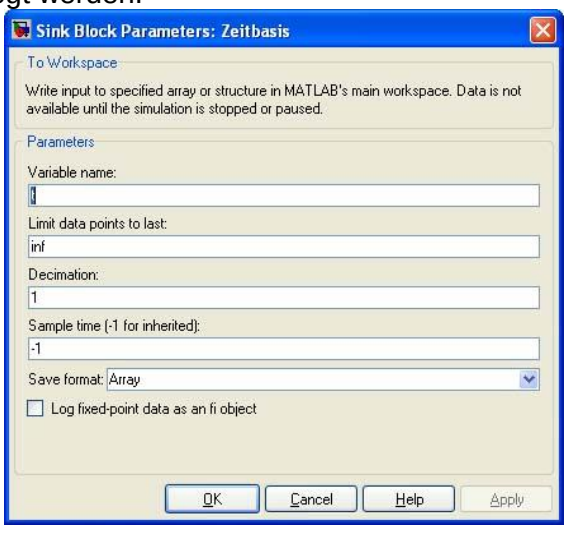
Jetzt wollen wir die 3 Blöcke parametrieren. Durch Doppelklick z. B. auf das erste Verzögerungsglied erhalten wir das Block-Parameter-Fenster und geben die Zeitkonstante T_1 = 1.6 Sek. ein. Ebenso verfahren wir mit dem 2. Verzögerungsglied (T_2 = 0.4 Sek). Man beachte, dass der **Dezimalpunkt kein Komma** sein darf, sondern **ein Punkt sein muss**.
Entgegen unserer Schreibweise (1 + s) wird bei **SIMULINK** (s + 1) geschrieben.
Die anderen Parameter ändern wir nicht.
Bei der Step-Funktion können wir mit **Step time** den Startzeitpunkt des Sprunges und mit **Final value** die Sprunghöhe einstellen.
Der Sprungfunktion kann ein Anfangswert (**Initial value**) gegeben werden, in unserem Beispiel 0.
Als Endwert (**Final value**) geben wir 1 ein (Einheitssprung).Die anderen Parameter werden in diesem Beispiel nicht verändert.
Wie rechts zu sehen ist, kann auch die Beschreibung der Blöcke geändert werden (1. PT 1 statt Transfer Fon 1). Einfach beim unmarkierten Block in den Text klicken und ändern. Weitere Formateinstellungen über die rechte Maustaste.



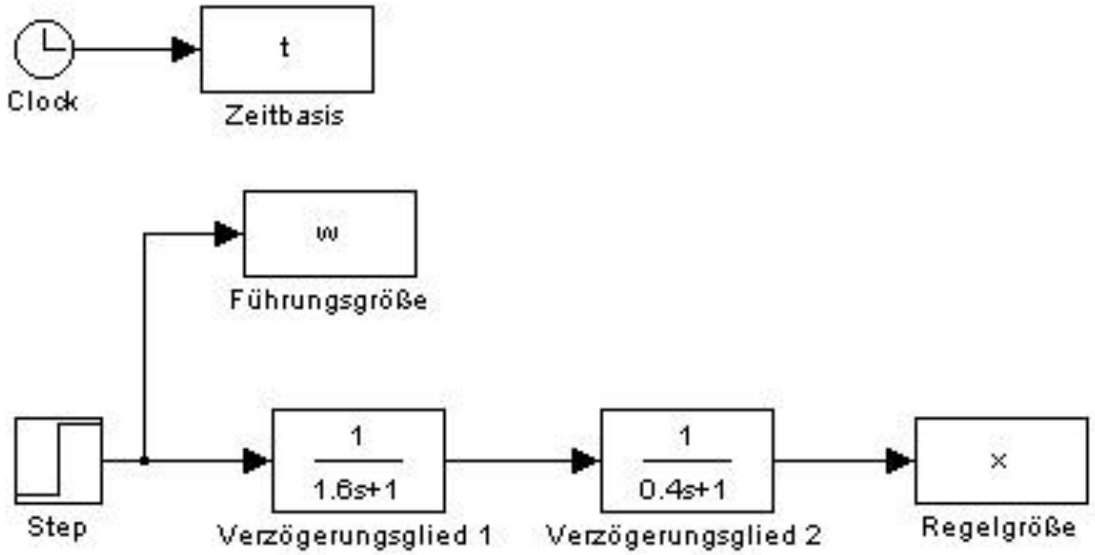
Nun muss noch geklärt werden, wie wir das Simulationsergebnis darstellen. In der "**Sink**" - Gruppe werden verschiedene Möglichkeiten angeboten. Unsere bevorzugte Methode ist das Schreiben des Simulationsergebnisses in einen Speicher, dem sog. "**To Workspace**".
Nachteil ist, dass wir immer nach der Simulation einen Plot-Aufruf im Matlab-Command-Fenster starten müssen. Vorteil ist aber, dass das dann gezeichnete Simulationsergebnis mehrfarbig dargestellt wird und so auf den Farbdrucker ausgegeben werden kann. Es kann auch als Bild gespeichert werden, um dieses in Dokumentationen einzufügen.



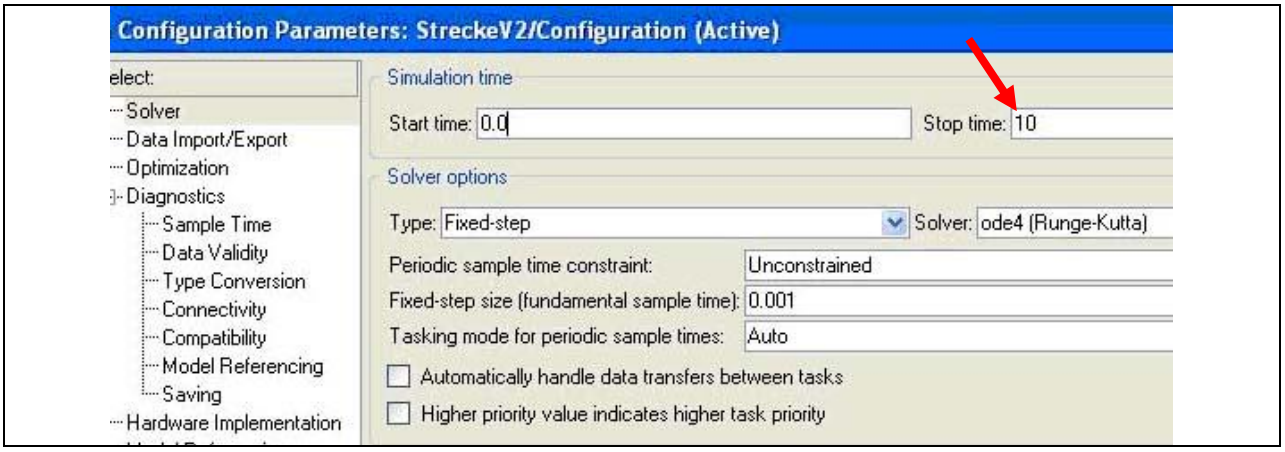
Ein Workspace ist letztlich nichts anderes als ein Speicherbereich (array), in dem die Simulationsergebnisse in Abhängigkeit des zeitlichen Verlaufs abgelegt werden.
Zur Darstellung über der Zeit benötigen wir deshalb noch einen "**Clock**"-Block aus der "**Sources**"-Gruppe, also einen Zeitgeber. Auch der Zeitverlauf muss in einem Workspace gespeichert werden.
Somit ergibt sich für diese Simulationsaufgabe ein Gesamtmodell (siehe unten), in dem wir auch den Führungssprung aufzeichnen wollen. Die Worskspaces sind von uns entsprechend ihrer Funktion (Zeitbasis, Führungsgröße, Regelgröße) umbenannt und mit eigenen Bezeichnungen für Variablen Namen (variable name) versehen. Im Bild rechts sehen wir das Fenster zum Ändern der Block-Para-meter. Statt dem ursprünglichen Variablen-Namen (simout) setzen wir unseren eigenen ein (t, w, x). Bei "**Save format**" muss noch "**Array**" ausgewählt werden



Gesamtmodell: Strecke mit 2 PT1 - Gliedern.

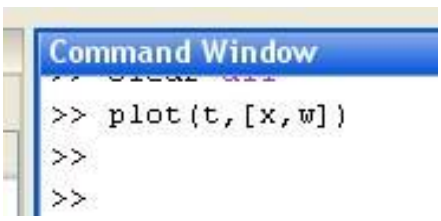


Jetzt sind nur noch die Simulationsparameter (siehe Bild) einzustellen. Da die größte Zeitkonstante 1,6 Sek. beträgt, sollte die Simulationszeit zunächst etwas mehr als 5-mal so groß gewählt werden (Start time = 0, Stop time = 10).

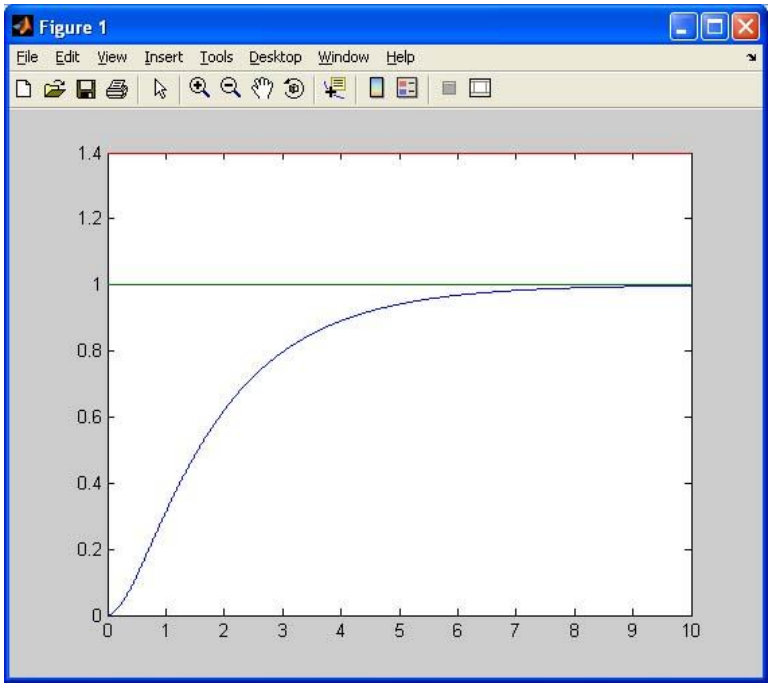


Die maximale und minimale Rechenschrittweite lassen wir wie eingestellt. Die Simulationszeit (Stop time) in s lässt sich auch in der Symbolleiste des Simulationsfensters einstellen (siehe Pfeil unten).
Nun können wir die Simulation starten. Wir haben 3 Möglichkeiten. Im Modell-Fenster (siehe Bild unten) mit dem "Startpfeil" (siehe Kreis) in der Mitte der Symbolleiste, mit "CTRL-T" oder im Pull down-Menü "Simulation" anklicken und dann auf "Start" klicken. Ein akustisches Signal zeigt das Ende der Simulation an.

Um nun das Simulationsergebnis grafisch darstellen zu können, muss im Matlab-Command-Fenster ein Plot-Aufruf gestartet werden. Wir wollen die Reaktion der Strecke auf einen Führungssprung sehen. Führungssprung (w) und die Antwort der Strecke (x) sollen dargestellt werden. Dazu das Plot-Kommando wie im Bild rechts **eingeben.**

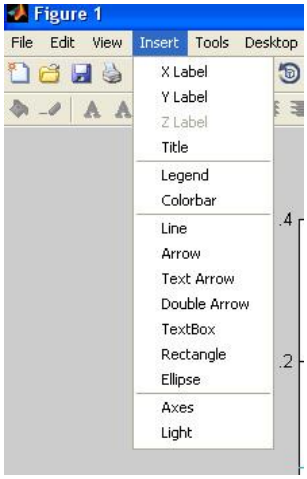


Nun öffnet sich ein neues Fenster (Figure 1) mit der grafischen Darstellung unseres Simulationsergebnisses (Bild rechts). Die darzustellenden Parameter schreiben wir in eckige Klammern. Der Aufruf lässt sich über die Pfeiltasten wiederholen.
Zur Darstellung weiterer Variablen kann das Plot-Kommando erweitert werden, z. B. **plot(t,[w,x,z])**.



Wie unter Windows üblich, kann dieses Bild auf dem angeschlossenen Farbdrucker oder über den Netzdrucker ausgegeben werden. Ebenso ist es auch möglich, das Bild selbst in 10 verschiedenen Formaten z. B. jpg oder bmp zu speichern und in ein Dokument zu kopieren.

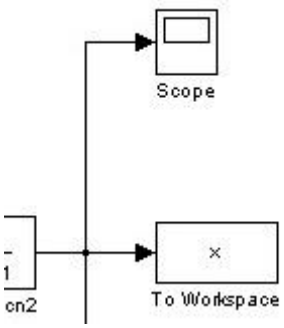
Zur besseren Dokumentation kann das Bild der hier gezeigten Plot-Darstellung editiert werden. Es ist somit möglich, die X-Y-Achsen zu beschriften, Titel einfügen, Textboxen einfügen, Legende beschriften, Data Cursor usw.

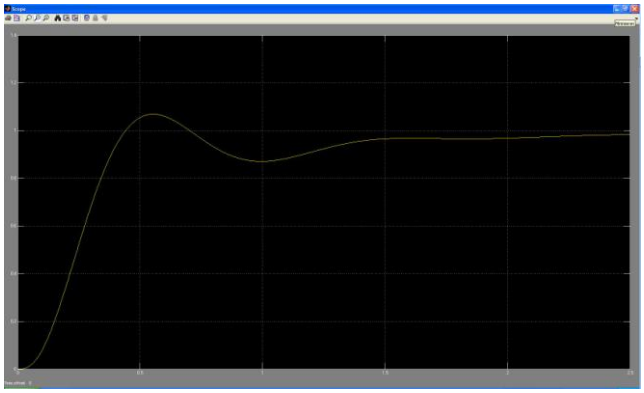


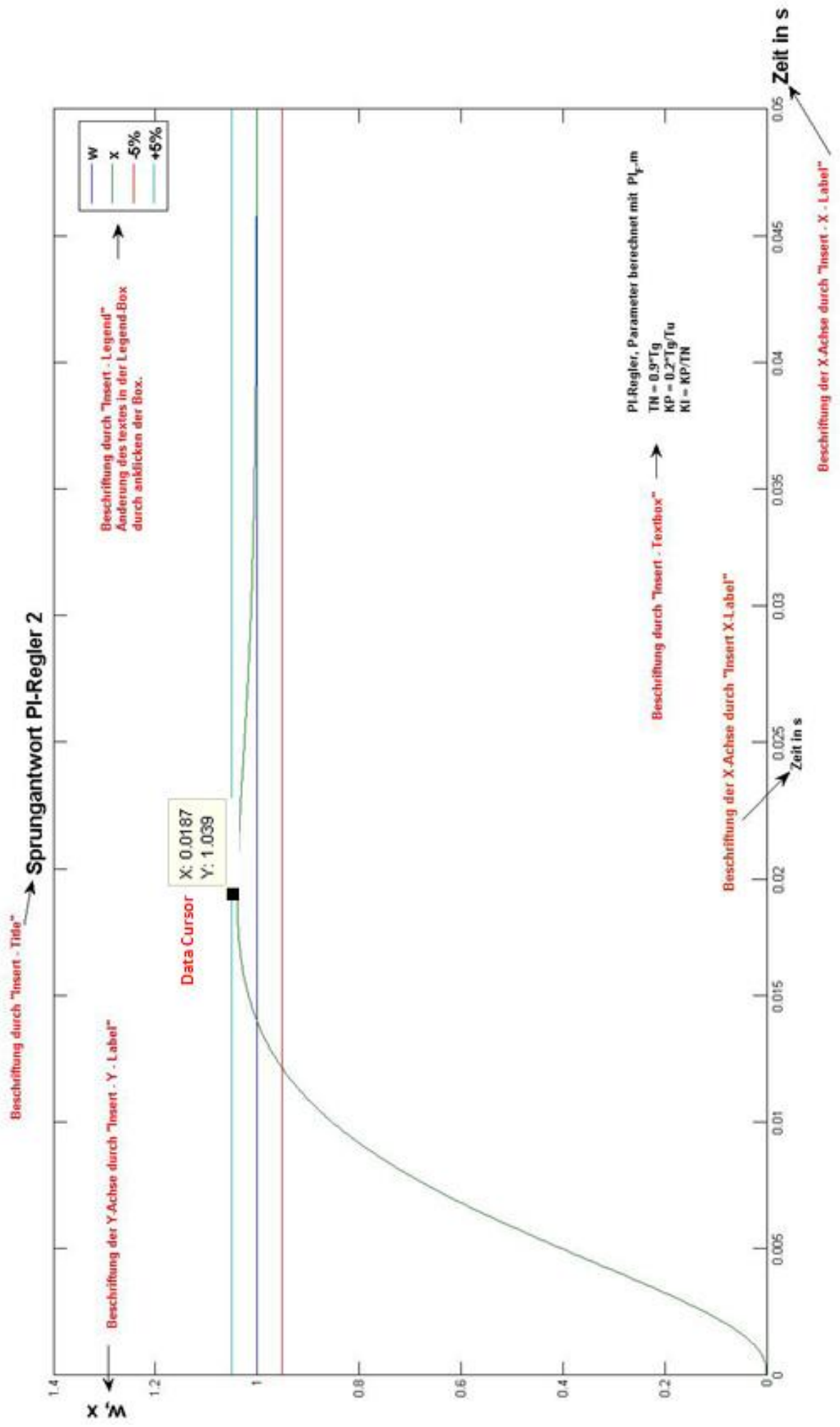
Wird über „**View**“ die „**Plot Edit Toolbar**“ aktiviert, stehen weitere Hilfsmittel zur Verfügung.

Hilfreiches finden wir auch unter „**Tools**“.

Das Simulationsergebnis kann auch mit dem „**Scope**“ aus der Gruppe „**Sinks**“ des Library-Browsers dargestellt werden. Auch hier ist es möglich, die Daten zu speichern.



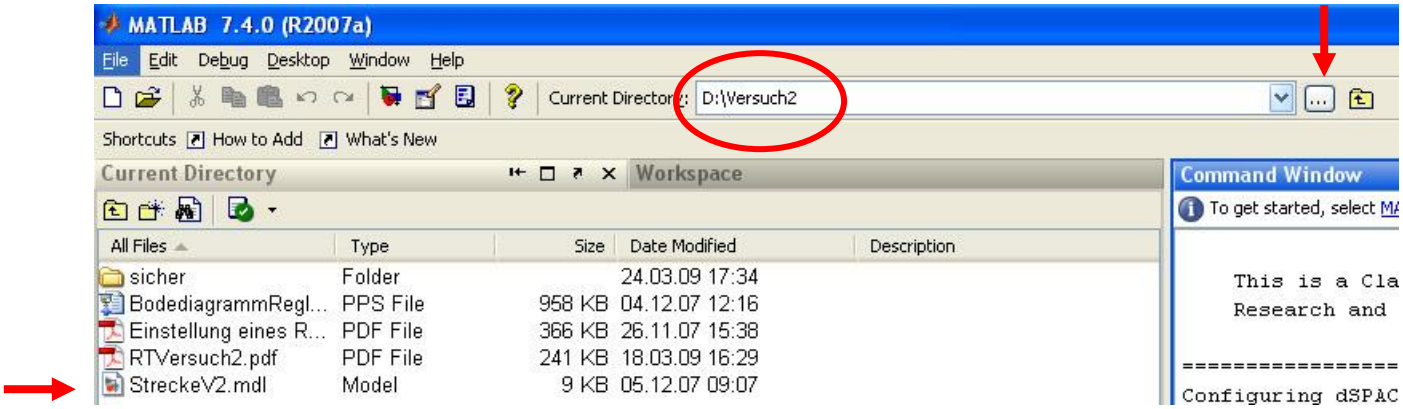



Sprungantwort PI-Regler 2
Beschriftung durch "Insert - Title"
Beschriftung der Y-Achse durch "Insert - Y - Label"
w, x
Zeit in s
Beschriftung der X-Achse durch "Insert X-Label"
Beschriftung der X-Achse durch "Insert - X - Label"
Data Cursor
X: 0.0187
Y: 1.039
Beschriftung durch "Insert - Legend"
Änderung des textes in der Legend-Box
durch anklicken der Box.
w
x
-5%
+5%
Beschriftung durch "Insert - Textbox"
PI-Regler, Parameter berechnet mit PI_s_m
TN = 0.9*Tg
KP = 0.2*Tg/Tu
KI = KP/TN

Formatieren von Textbox usw. durch Markieren, klick mit rechter Maustaste. Im Pull-Down-Menü werden diverse Möglichkeiten angeboten, z. B. „Show Property Editor usw.

Wenn Sie das Simulationsbild vor weiteren Anwendungen nicht löschen, bleiben Ihre Textboxen, Pfeile usw. erhalten und können passend zum neuen Ergebnis editiert werden.
Titel und X-Y-Labels werden nicht übernommen.

Um Ihnen den Einstieg zu erleichtern empfiehlt es sich, die in den Aufgaben empfohlenen Verzeichnisse zu wählen wie z. B. bei Versuch 2. Die zu regelnden Strecken sind gewöhnlich da schon vorhanden und sollen auch verwendet werden.



Diese wenigen Informationen reichen zum Arbeiten mit einfachen Modellen aus. Alles Weitere lernt man im Umgang mit dem Programm.

M-Files

Die zu ermittelnden Reglerparameter kann man sich mit den bekannten Methoden auch von Matlab berechnen lassen. Dazu verwendet man sogenannte m-files. Sie sind im ***Matlab*-Command-Window** lauffähig. Man spart sich dabei Berechnungen mit dem Taschenrechner und hat eine Dokumentation der Berechnungen, was ein großer Vorteil ist.

Haben wir z. B. bei einem PID-Regler $\mathbf{K_I}$ aus dem Bodediagramm erhalten, kann man zur Berechnung von $\mathbf{K_P}$ und $\mathbf{K_D}$ wie folgt vorgehen:

Mit dem in Matlab verfügbaren Editor erstellt man die Algorithmen und speichert die Datei mit einem passenden, frei gewählten Namen, z. B. PID_Bode.m ab. Die zur Berechnung erforderlichen Konstanten (TR1 usw.) müssen definiert werden.
Das m-File (= Programm) kann auch schon während der Versuchsvorbereitung Zuhause einfach in einem Editor oder WORD erstellt werden.

Hier ein Beispiel (vor Kommentare wird einfach das %-Zeichen gesetzt, wie // bei C):

```
% m-file zur Ermittlung von KP und KD , wenn KI aus dem Bodediagramm bekannt ist

clear all                    % alle früher berechneten Parameter löschen

TR1=1.2
TR2=0.8
KI=3.9                       %  KI aus Bodediagramm
KP=KI*(TR1+TR2)
TV=TR1*TR2/(TR1+TR2)
KD=KP*TV
Test1                        % eventuell Aufruf des zugehörigen Modells, hier: Test1.mdl,
```

Somit sind alle Reglerparameter definiert, wenn diese Datei in der „**Current Folder**“ markiert und mit F9 gestartet wird.

Start der Berechnung auch durch klick auf  . Die Rechenergebnisse werden im Matlab Command Window angezeigt.
Auf Wunsch kann durch Eingabe des Namens des Simulink Modells (hier Test1) dieses aufgerufen werden. Es erscheint dann auf dem Bildschirm.

Im Modell selbst werden nicht die berechneten Zahlenwerte eingegeben sondern nur die Variablennamen **KP, TV und KD**.

Wichtig! **Im Modell unbedingt die gleiche Schreibweise der Variablen verwenden wie im m-File.**

Ein weiterer Vorteil der Verwendung von m-Files besteht darin, dass beim Testen mit verschiedenen Parametern diese nur im m-File geändert werden müssen. Nach dem Neustart des m-Files sind die Blockparameter im Modell aktualisiert. Gleichzeitig findet eine Dokumentation statt. Nach erneuter Simulation und Plot steht das neue Ergebnis zur Verfügung.

Weiteres Beispiel: $$\mathbf{K_p = \frac{2 \cdot D \cdot T_S \cdot \pi - T_m \cdot \sqrt{1 - D^2}}{K_S \cdot T_m \cdot \sqrt{1 - D^2}}}$$

```
clear all                    % früher berechnete Parameter löschen

    D=0.7
    TS=12
    Tm=20
    KS=4
    KP=(2*D*TS*pi - Tm*sqrt(1-D^2))/(KS*Tm* sqrt(1-D^2))

    % pi ist dem System bekannt mit pi = 3,1416
```

Somit ist der Parameter **KP** definiert, wenn dieses m-File gestartet wird.
Achtung: Mit den Klammern gut aufpassen!

Man kann das m-File noch so erweitern, dass die Simulation des Models automatisch gestartet und das Ergebnis geplottet wird. Man erweitert das Programm wie folgt: Hochkommata nicht vergessen!):

```
sim('Laufwerk:\Verzeichnis\Programmname.mdl')

plot(t,[w,x])   % also plot(Zeitbasis,[Augabe1,Ausgabe2,…])
```

MATLAB-SIMULINK ist sehr mächtig. Die Laborversuche sind nur ein sehr bescheidener Einstieg. Diese Anleitung beschreibt nur einen ganz kleinen Teil dessen, was mit **MATLAB SIMULINK** alles möglich ist.