

# Software: SimX - Einfuehrung - DC-Motor

Aus OptiYummy

↑

← →

**Simulation - eine praktische Einfuehrung**  
**Autor: Dr.-Ing. Alfred Kamusella**



- ohne Altersbeschränkung -



Dieses Beispiel wurde ursprünglich aufbereitet, um Gymnasial-Schülern bei einem Besuch an der Universität durch selbstständiges, praktisches Üben das Erlebnis der numerischen Simulation zu vermitteln. Dazu wurde als Objekt ein Antrieb mit einem Gleichstrom-Motor gewählt.

Im Rahmen einer studentischen Komplexübung zur Dimensionierung eines Nadelantriebes für einen Brailleschrift-Präger dient dieses Beispiel zur Einarbeitung in das Bedienkonzept von SimulationX.

## 1. SimulationX - ein Simulationsprogramm

- Oberflächlicher Einblick

## 2. Modell eines elektrischen Antriebs

- Motor mit Netzteil
- Einschaltstrom - Experiment
- Drehzahl - Experiment
- Regelkreis - Experiment

## 3. Was ist nun eigentlich Simulation?

- Eine erschöpfende Antwort

← →

Von „[http://www.optiyummy.de/index.php/Software: SimX - Einfuehrung - DC-Motor](http://www.optiyummy.de/index.php/Software:_SimX_-_Einfuehrung_-_DC-Motor)“

---

# Software: SimX - Einfuehrung - DC-Motor - Programm

Aus OptiYummy

↑

← →

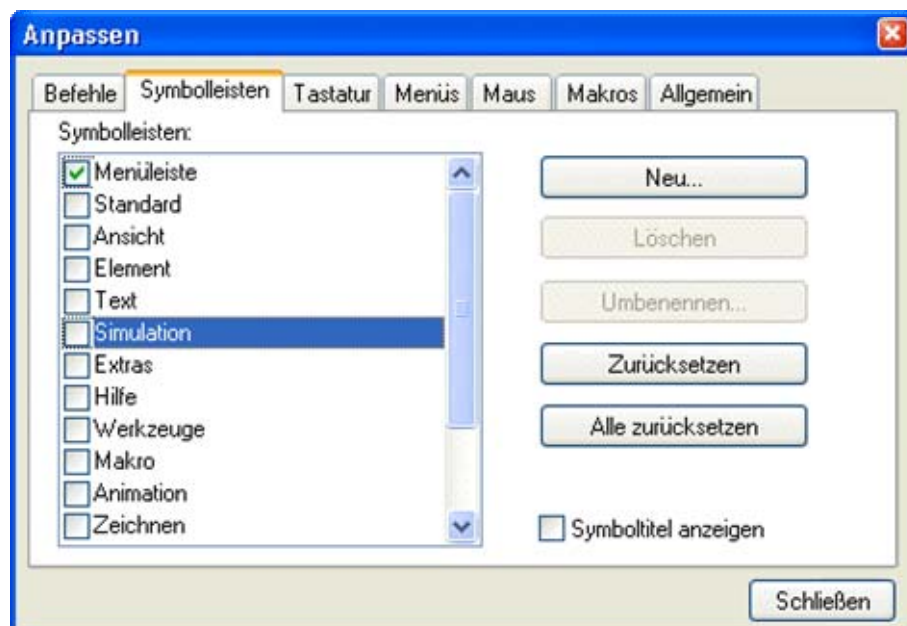
## SimulationX - ein Simulationprogramm

- Das von der Firma ITI GmbH aus Dresden entwickelte Simulationssystem gehört zu den Modernsten (Details findet man in der Produktbeschreibung).
- Wir wollen SimulationX in dieser praktischen Einführung nutzen, um zu demonstrieren, wie das mit dem Simulieren auf dem Computer so funktioniert.
- Das Programm steht als freie Studenten-Version mit eingeschränktem Funktionsumfang zur Verfügung. Teilnehmer einer Einführungsveranstaltung am Institut für Feinwerktechnik und Elektronik-Design können es als kleines "Werbegeschenk" beim Betreuer erhalten. Das Programm darf an Bekannte weitergegeben werden.
- Auswärtige Interessenten können diese Demo-Version von SimulationX bei der Firma ITI direkt anfordern bzw. herunterladen.



Nun geht es los!

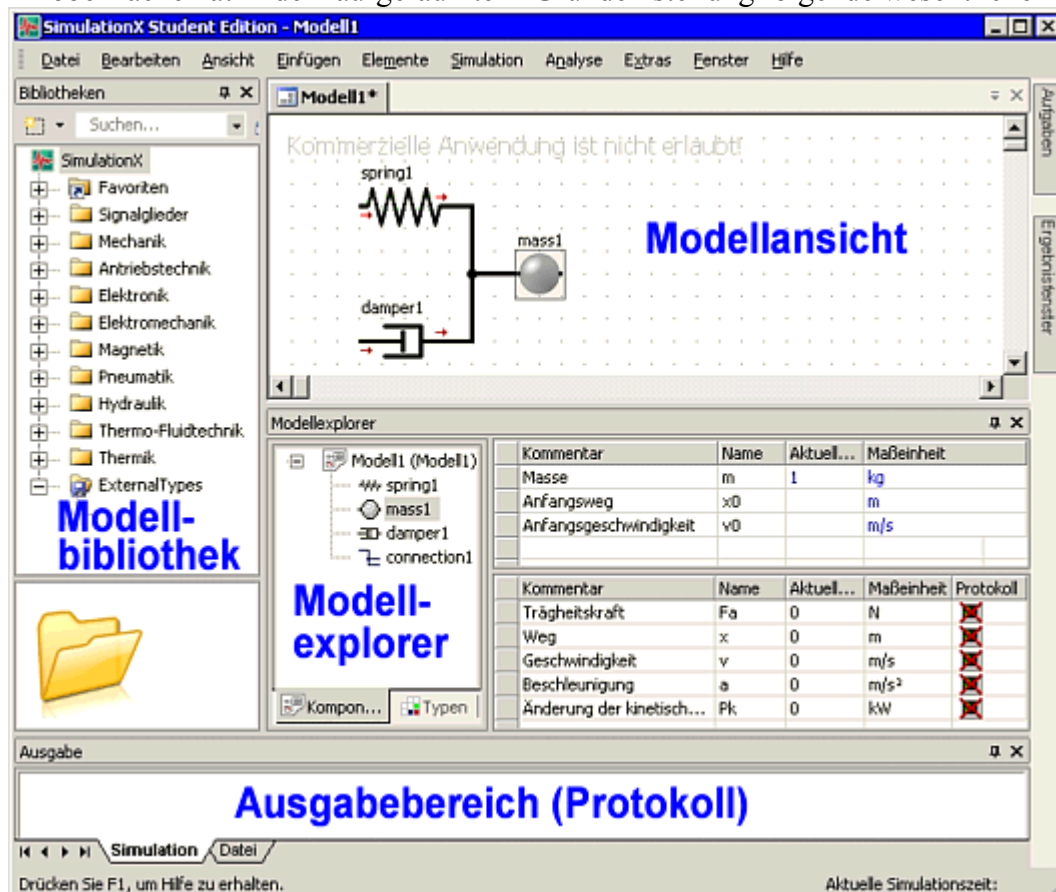
- Wir starten SimulationX. Nach dem Start ist bereits ein leeres Modell geöffnet.
- **Achtung:** Benutzeroberfläche anpassen!
  - Hat man noch keine individuellen Anpassungen der Benutzeroberfläche vorgenommen, so wirkt diese infolge einer Vielzahl von Werkzeugleisten sehr überladen.
  - Deshalb sollte man unbedingt mittels *Extras* > *Anpassen* alle Symbolleisten außer der Menüleiste ausblenden:



- Günstig ist es, in diesem Zusammenhang auch die Menüs immer vollständig anzeigen zu lassen:



- Die Programmoberfläche hat in der "aufgeräumten" Grundeinstellung folgende wesentlichen Bereiche:



- In der **Leiste der Modellbibliothek** findet man ein reichhaltiges Angebot an Modell-Elementen aus allen möglichen Fachgebieten:



- In den **Bereich für die Modellansicht** kann man diese Elemente einfach mit dem Cursor ziehen:
  - Wie in einem "echten" Versuchsstand kann man die Elemente miteinander verbinden.
  - Zum Verbinden nutzt man wieder den Cursor.
  - Das gelingt aber nur, wenn die Element-Anschlüsse zueinander passen (auch in der wirklichen Welt funktioniert eine elektrische Leitung meist nicht besonders gut als Wasserleitung!)
- Im **Modell-Explorer** erscheint das aufgebaute Modell als Baumstruktur:
  - Die Form dieser Darstellung entspricht der Ordneransicht im Windowsexplorer.
  - In dieser Baumstruktur werden die Elemente und die Verbindungen abgebildet.
  - Wenn man ein Element in der Baumstruktur oder in der Modellansicht auswählt, erscheinen die zugehörigen Informationen rechts neben der Baumstruktur.
- Im **Ausgabebereich** erscheinen z.B. Fehlermeldungen bei der Arbeit mit dem Modell (hoffentlich nicht!).



Von „<http://www.optiyummy.de/index.php/Software: SimX - Einfuehrung - DC-Motor - Programm>“

# Software: SimX - Einfuehrung - DC-Motor - Antriebsmodell

Aus OptiYummy

↑

← →

## Modell eines elektrischen Antriebs

Es gibt nur wenige elektrische Geräte, die keinen elektrischen Antrieb enthalten (z.B. einfache Wasserkocher oder Leuchten). Ansonsten findet man fast überall mindestens einen kleinen Motor versteckt.

- Oft reicht es nicht aus, dass sich der Motor dreht:
  - der Motor soll z.B. zu jedem Zeitpunkt die richtige Drehzahl besitzen;
  - zu bestimmten Zeitpunkten soll sich der Motor z.B. um einen bestimmten Winkel gedreht haben;
  - Motoren dürfen z.B. nicht zu schnell beschleunigen oder bremsen, damit nichts kaputt geht;
  - es muss manchmal schneller gehen als man gucken kann, z.B. in Bruchteilen von Sekunden und
  - der Antrieb darf meist fast nichts kosten!
- Das Konstruieren eines modernen Antriebssystems ist meist eine anspruchsvolle Aufgabe:
  - So lange in der Werkstatt basteln, bis es funktioniert - das funktioniert nicht oder dauert viel zu lange!
  - Außerdem kostet jede Stunde Arbeit in der Werkstatt ungefähr 100 €. Das kann keiner bezahlen.
  - Deshalb entwickelt man zuerst numerische Modelle, mit denen man auf dem Computer eine Lösung sucht.
  - Erst wenn man weiß, wie man den Antrieb aufbauen muss, lässt man sich ein Versuchsmuster in der Werkstatt bauen.
- An einem vereinfachten Lehrbeispiel soll nun das Arbeiten mit solchen Computer-Modellen gezeigt werden.

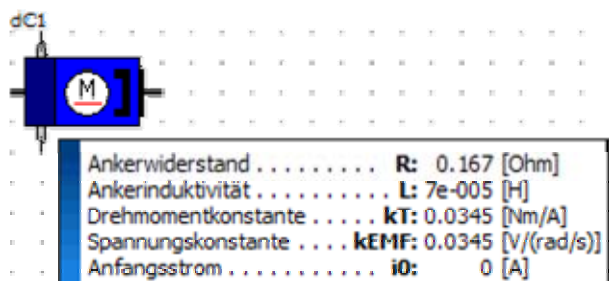
Damit es nun richtig losgehen kann, müssen wir zuvor unseren Modell-Ansichtsbereich entrümpeln, falls wir zum Ausprobieren dort ein paar Modell-Element verknüpft hatten:

- Die Modell-Elemente kann man einfach löschen.
- Oder man schließt das Modell (*Datei* > *Schließen*) und öffnet ein neues Modell (*Datei* > *Neu*).



**Gleichstrom-Motor**

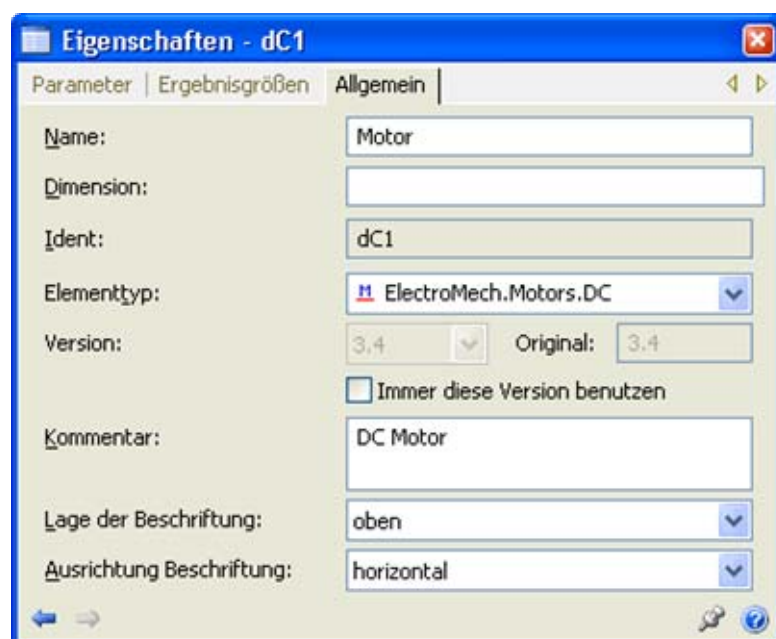
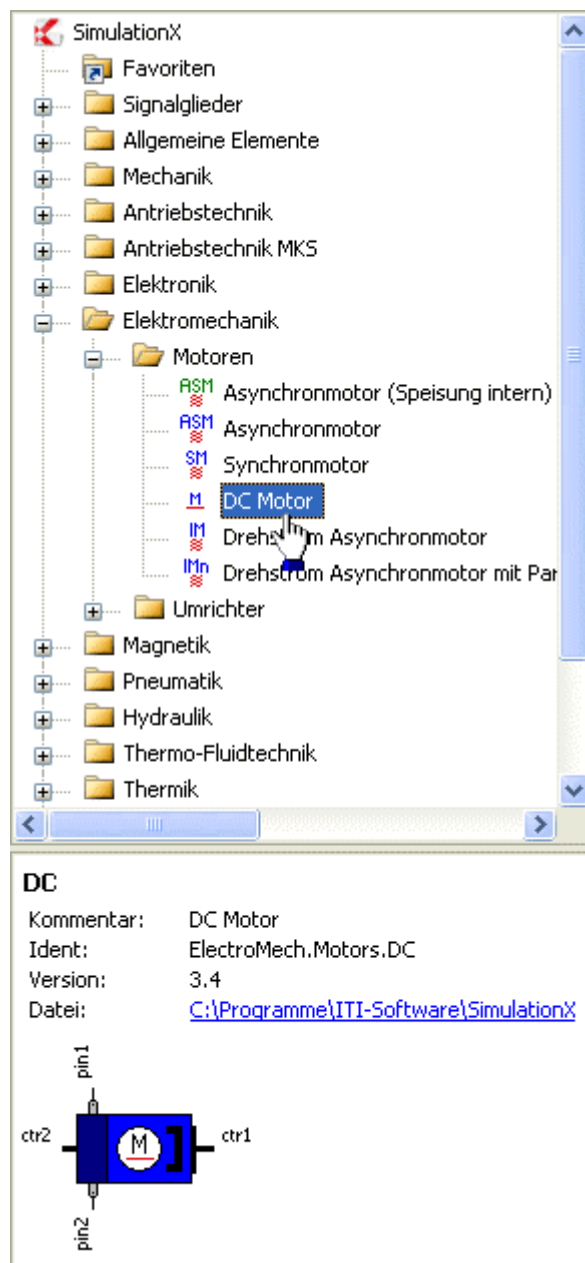
- Den Aufbau unseres Modells wollen wir mit dem Motor beginnen.
- Dazu zieht man dieses Modell-Element einfach aus der Bibliotheksleiste auf die Arbeitsfläche (Modellansicht-Bereich).
- Bewegt man danach den Cursor über dieses Modell-Element, erscheint ein kleines Fenster mit den aktuellen Parameterwerten:



- Unser Motor, den wir für den Antrieb benutzen möchten, besitzt natürlich ganz andere Parameter. Nach einem Doppelklick auf das Motor-Symbol öffnet sich ein Eigenschaftsfenster.
- Hier tragen wir die "richtigen" Parameter ein. Dabei muss man beachten, dass man die richtige Maßeinheit gewählt hat (z.B. bei der Ankerinduktivität **H** umgeschaltet auf **mH**):



- Unser Motor erhielt automatisch den Namen **dC1**. In der Register-Karte *Allgemein* des Eigenschaftsdialogs kann dafür eine günstigere Bezeichnung eintragen:

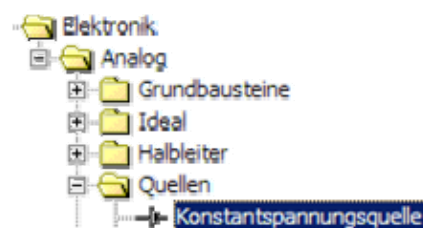
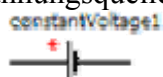


- Die Änderung des Element-Namens ist auch direkt in der grafischen Modellansicht möglich.
- Nach dem Einschalten des Motors wollen wir uns den Signalverlauf des elektrischen Stromes anschauen, der dann durch die Ankerspule des Motors fließt. In der Register-Karte *Ergebnisgrößen* kann man die Signale Freischalten, welche man sich anschauen möchte:



### Stromversorgung (Netzteil)

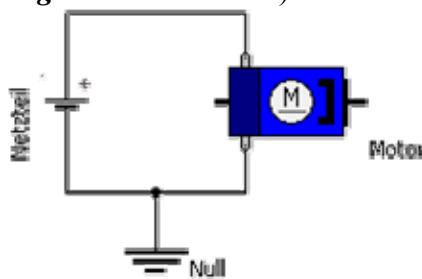
- Zum Einschalten des Motors benötigt man noch ein Netzteil. Der Motor soll mit 24 V betrieben werden. Wir benutzen aus der Bibliothek eine Konstantspannungsquelle.



- Die Lage des Symbols ist noch ungünstig für die Verbindung mit dem Motor. Die Elemente kann man jedoch noch beliebig verschieben und drehen. Zum Drehen nutzt man nach dem Anwählen des Elements die Funktion *Elemente > Nach rechts drehen*. Das Verschieben erfolgt einfach mit dem Cursor.



- Der Name der Spannungsquelle soll **Netzteil** heißen. Die Lage der Beschriftung kann man in der Registerkarte *Allgemein* der Element-Eigenschaften ändern (im Beispiel auf "links" und "vertikal").
- Das Netzteil soll eine Spannung von 24 V liefern.
- Die Anschlüsse des Netzteils sollen mit den Anschlüssen des Motors verbunden werden (einfach mit der linken Maustaste!).
- Um die Schaltung mit einem Null-Potential zu versehen, benötigen wir noch eine elektrische Masse (in der Bibliothek unter *Elektronik-Analog-Grundbausteine*).



- Auf der folgenden Seite wird beschrieben, wie wir in einem ersten Experiment den Anstieg des Motorstroms nach dem Anlegen der Betriebsspannung beobachten können.

*Datei > Speichern* nicht vergessen: **Antrieb.ism**



Von „[http://www.optiyummy.de/index.php/Software: SimX - Einfuehrung - DC-Motor - Antriebsmodell](http://www.optiyummy.de/index.php/Software:_SimX_-_Einfuehrung_-_DC-Motor_-_Antriebsmodell)“

# Software: SimX - Einfuehrung - DC-Motor - Einschaltstrom

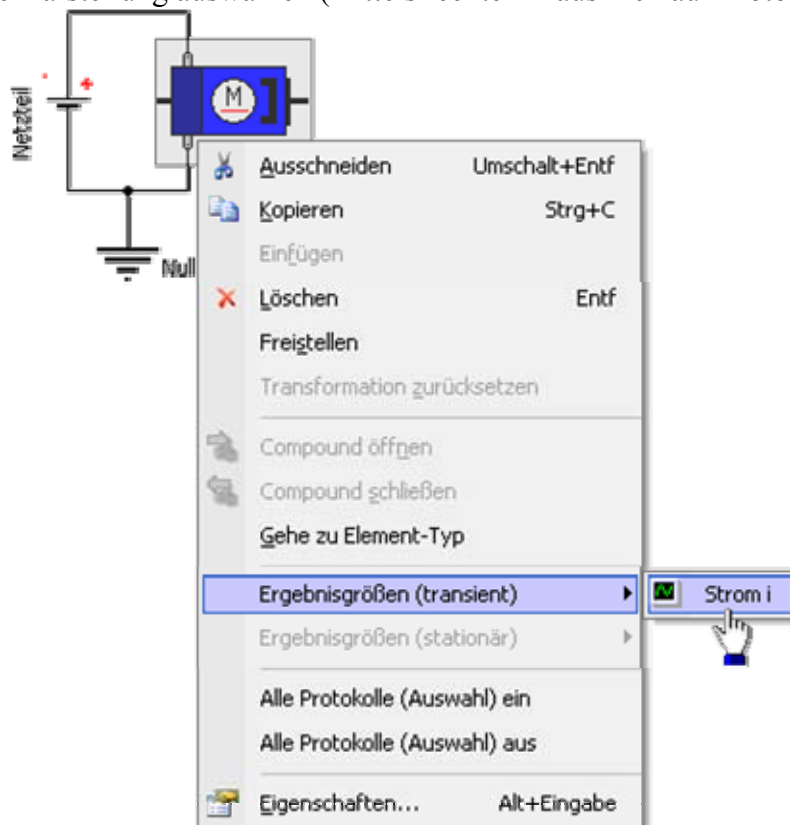
Aus OptiYummy

↑

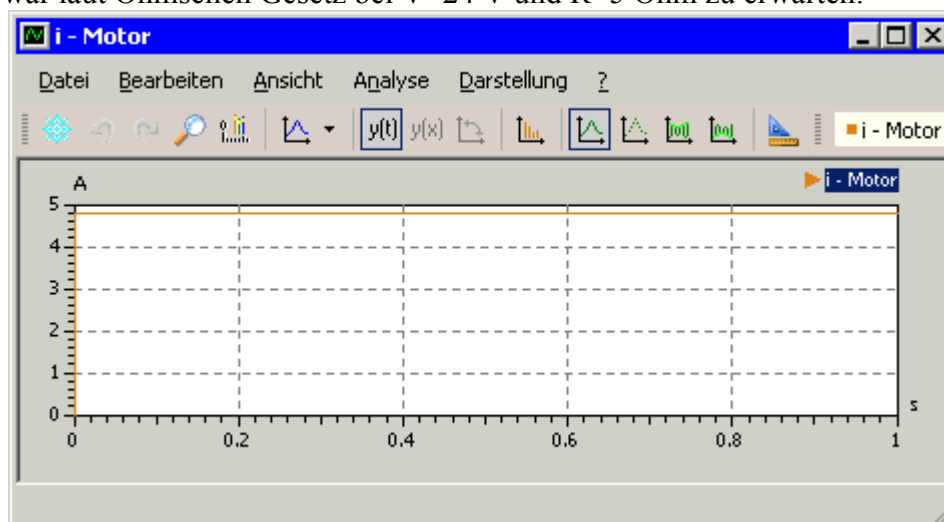
← →

## Einschaltstrom-Experiment

- Damit man der Verlauf des Motorstroms nach dem Einschalten betrachten kann, muss man zuvor diese Ergebnisgröße für die Darstellung auswählen (mittels rechten Mausklick auf Motor):



- Es wird ein leeres Ergebnisfenster für den Motor-Strom geöffnet, da wir noch keine Simulation gestartet hatten.
- Wir starten einfach mal einen Simulationslauf, um zu sehen, was passiert (**Simulation > Start**).
- Im Ergebnisfenster wird nun ein konstanter Strom von 4,8 A im Zeitbereich von 0 bis 1 Sekunde angezeigt. Das war laut Ohmschen Gesetz bei  $V=24\text{ V}$  und  $R=5\text{ Ohm}$  zu erwarten:



- Im Motor befindet sich jedoch eine Spule, welche eine bestimmte Induktivität besitzt. Deshalb dürfte der Strom nicht "schlagartig" seinen Endwert erreichen, sondern muss vom Wert=0 A kontinuierlich ansteigen.
- Der standardmäßig simulierte Zeitbereich von 1 s ist für unsere Spule viel zu groß! Die Zeitkonstante **T**

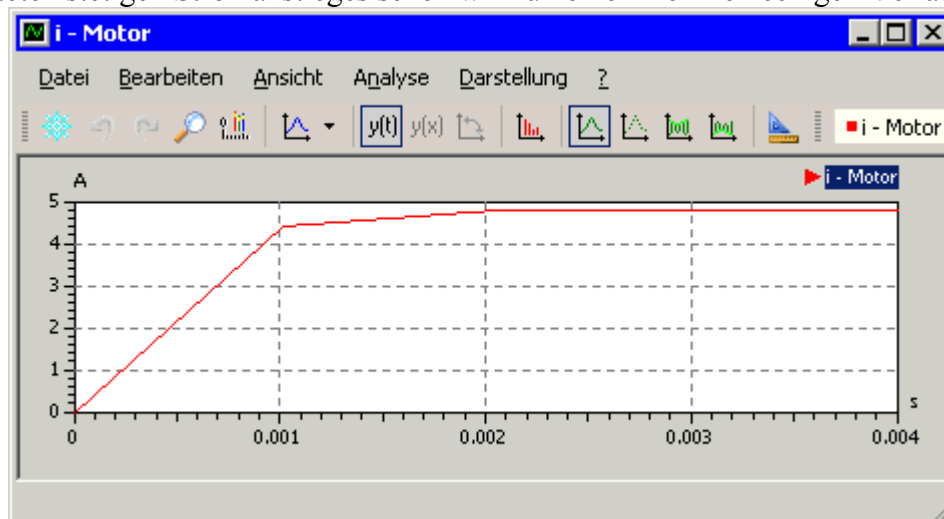
für den Stromanstieg beträgt

$$T=L/R=0.002/5 \text{ s}=0.4 \text{ ms}$$

- Da nach ungefähr der dreifachen Zeitkonstante der Endwert des Stromes fast erreicht ist, genügt eine Simulationszeit von **tStop=4 ms**.

**Simulation > Einstellungen Transient** gestattet neben vielen anderen Einstellungen auch die Festlegung des berechneten Zeitbereiches:

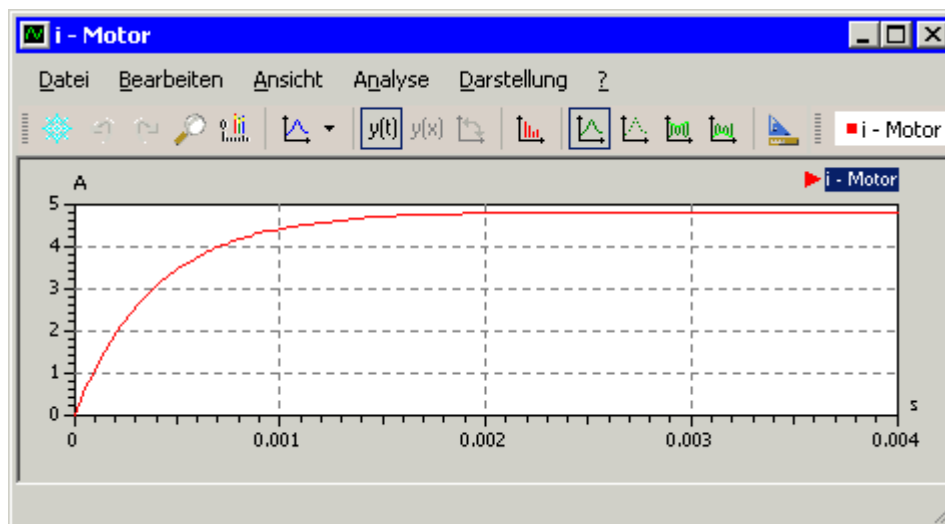
- Wir verringern in der Registerkarte "Allgemein" **tStop=4 ms**.
- Nach **Simulation > Rücksetzen** starten wir erneut einen Simulationslauf.
- Statt des erwarteten stetigen Stromanstieges sehen wir nun einen ziemlich eckigen Verlauf:



- Ursache ist die "minimale Ausgabeschrittweite" **dtProtMin=1 ms**. Die Stromkurve wird demzufolge im Beispiel aus Geradenstücken von 1 ms Länge approximiert!
- Damit Kurvendarstellungen unabhängig vom Zeitbereich einigermaßen "rund" sind, sollte man die Ausgabeschrittweite abhängig vom berechneten Zeitbereich machen. Für die Max. Rechenschrittweite ist z.B. standardmäßig **dtMax=(tStop-tStart)/100** eingetragen.
- So soll auch die Ausgabeschrittweite angepasst werden: **dtProtMin=(tStop-tStart)/100**.
- Zusätzlich sollte die Protokollierung von Ergebnissen "**Nach mindestens dtProtMin sowie vor und nach Ereignissen**" erfolgen. Damit werden auch Unstetigkeiten in Signalverläufen "exakt" abgebildet:

Allgemein		Rücksetzpunkte	Tracing	Solver
Startzeit	tStart:	0	s	
Stopzeit	tStop:	0.004	s	
Min. Rechenschrittweite	dtMin:	1e-008	s	
Max. Rechenschrittweite	dtMax:	(tStop-tStart)/100	s	
Absolute Toleranz	absTol:	1e-005	-	
Relative Toleranz	relTol:	1e-005	-	
Minimale Schrittweite	dtDetect:	dtMin*1e-4	s	
Protokollierung von Ergebnissen:				
Nach mindestens dtProtMin sowie vor und nach Ereignissen				
Min. Ausgabeschrittweite	dtProtMin:	(tStop-tStart)/100	s	
Rücksetzen				

- Nun müsste nach erneuter Simulationsrechnung ein knickfreier Stromverlauf sichtbar sein:



- Im folgenden Experiment werden wir die Drehung des Motors und die Rückwirkung der Motor-Drehung auf den zeitlichen Verlauf des Stroms untersuchen.

← →

Von „<http://www.optiyummy.de/index.php/Software: SimX - Einfuehrung - DC-Motor - Einschaltstrom>“

# Software: SimX - Einfuehrung - DC-Motor - Drehzahl

Aus OptiYummy

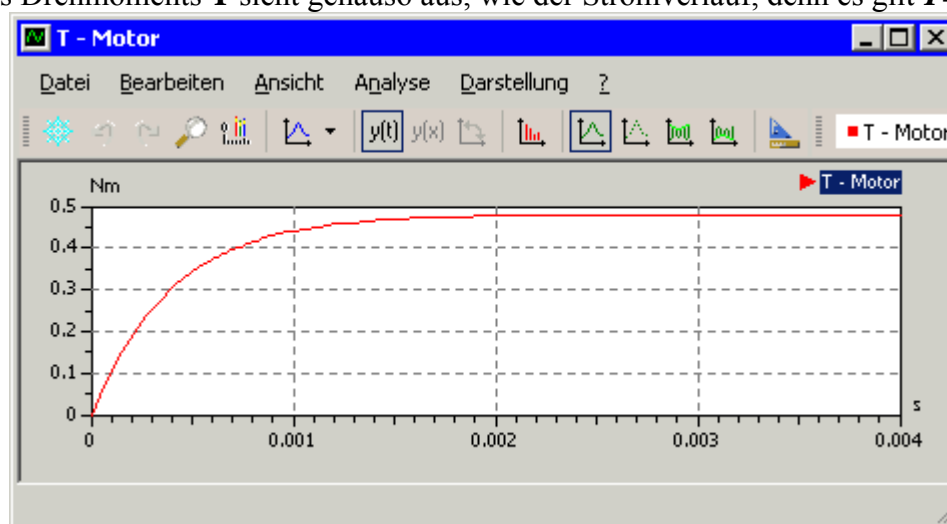
↑

← →

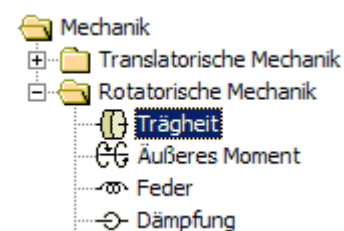
## Drehzahl-Experiment

Wenn ein elektrischer Strom durch die Spule des Motors fließt, wird ein Drehmoment erzeugt und der Motor beginnt sich zu drehen. Diesen Vorgang werden wir jetzt mit dem Modell simulieren. Zusätzlich zum Strom wollen wir das Drehmoment des Motors als Signalverlauf darstellen:

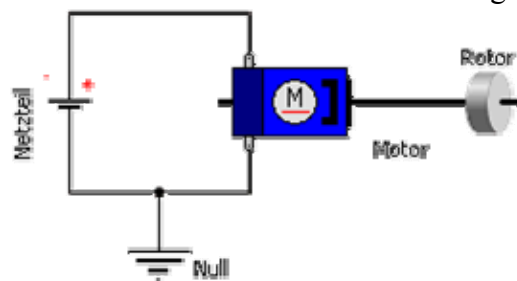
- Luftspaltpmoment **T** als Ergebnisgröße des Motors für die Ausgabe aktivieren.
- Für den Motor dieses Luftspaltpmoment **T** in einem neuem Signalfenster darstellen.
- Simulation zurücksetzen und starten, damit Signalverlauf berechnet wird.
- Der Verlauf des Drehmoments **T** sieht genauso aus, wie der Stromverlauf, denn es gilt  $T=kT \cdot i$ .



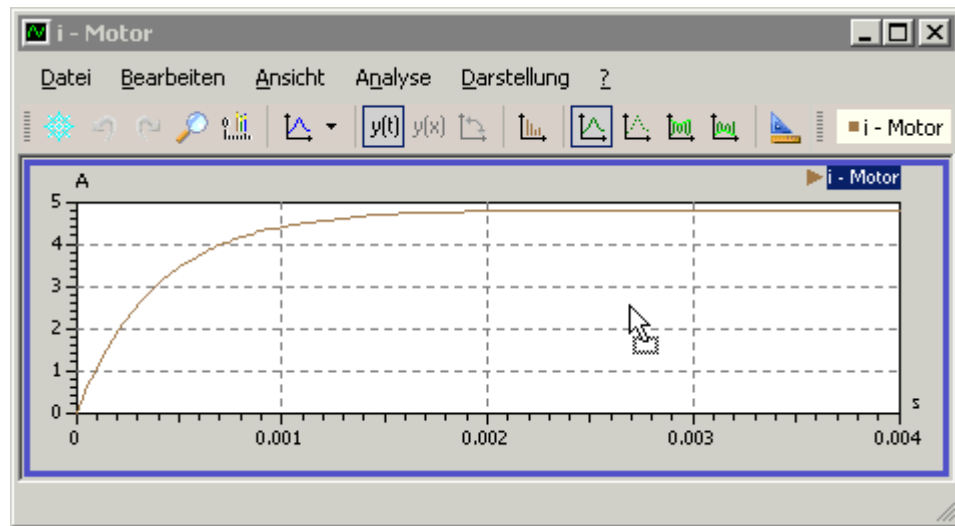
Das in der Bibliothek bereitgestellte Motor-Modell besitzt keine beweglichen, drehbaren Teile! Damit sich etwas drehen kann, muss man also noch die drehbare Masse des Rotors von  $40 \text{ g} \cdot \text{cm}^2$  als Modell-Element ergänzen:



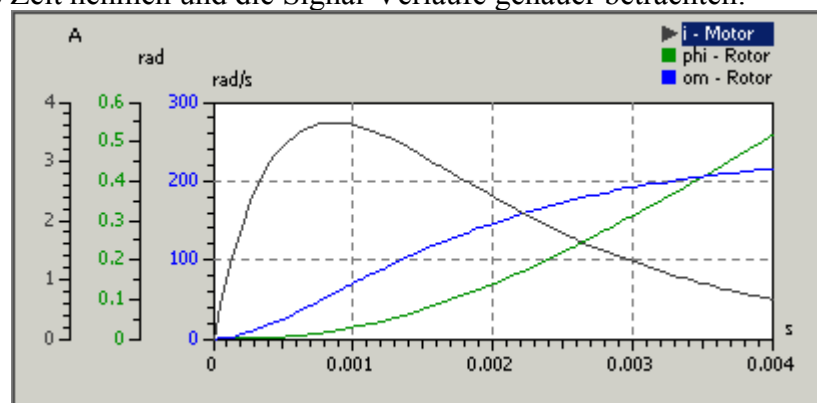
- In der Bibliothek existiert eine physikalische Domäne "Rotatorische Mechanik". Hieraus verwenden wir die rotatorische Trägheit:



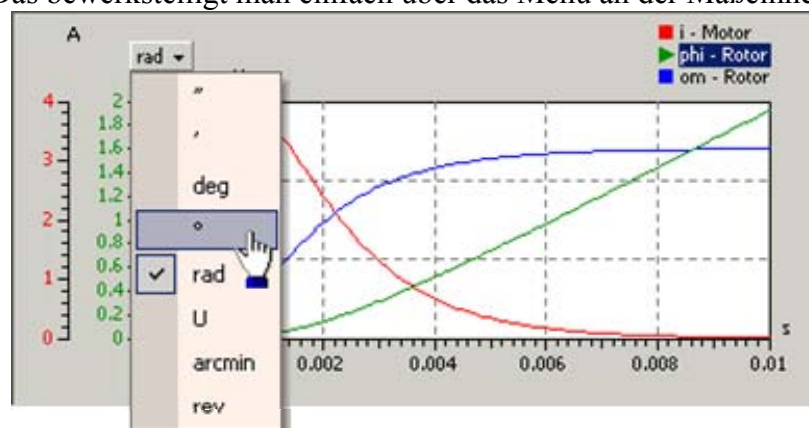
- Der Rotor unseres Motors besitzt eine Drehträgheit  $J=40 \text{ g} \cdot \text{cm}^2$ .
- Wir wollen uns in Form von Signalen anschauen, um welchen Winkel **phi** sich der Rotor dreht und wie groß die Drehgeschwindigkeit **om** wird.
- Wir haben inzwischen 4 Signalfenster auf dem Bildschirm, so dass es langsam etwas unübersichtlich wird:
  - Das Signal-Fenster **T-Motor** können wir schließen.
  - Die übrigen Signale ziehen wir mit dem Cursor per *Drag&Drop* einfach in das Signal-Fenster für den Strom (Hinweis: Ziehen der Signal-Legende, z.B. **om - Rotor**).
  - Im Zielfenster deutet beim *Drag&Drop* ein farbiger Rahmen an, ob das vorhandene Diagramm genutzt wird, oder ob ein neues Diagramm angelegt wird:



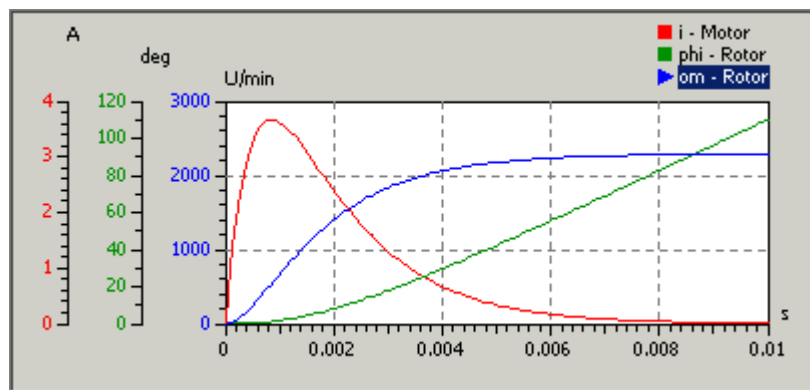
Nun wollen wir uns die Zeit nehmen und die Signal-Verläufe genauer betrachten:



- Der Strom  $i$  steigt wieder von Null beginnend an. Aber anstatt einen Endwert von 4,8 A zu erreichen, verringert sich der Motor-Strom dann wieder.
- Ursache für das Absinken des Motorstroms ist die beginnende Drehbewegung des Motors. Die Rotor-Drehzahl  $\omega_m$  erhöht sich stetig.
- Der vom Rotor zurückgelegte Winkel  $\phi$  wird dabei immer größer.
- Am Ende der durchgeführten Simulation haben die Signale noch keinen Endwert erreicht. Deshalb wollen wir in der Simulationssteuerung **tStop=10 ms** setzen. Ohne vorheriges *Rücksetzen* kann man die Simulationsrechnung durch *Start* beginnend von der bisherigen Endzeit 4 ms fortsetzen lassen.
- Anstatt der Maßeinheit **rad** wollen wir die anschaulichere Einheit  $^\circ$  verwenden (Hinweis: das Grad  $^\circ$  ist identisch mit **deg**). Das bewerkstelligt man einfach über das Menü an der Maßeinheit:

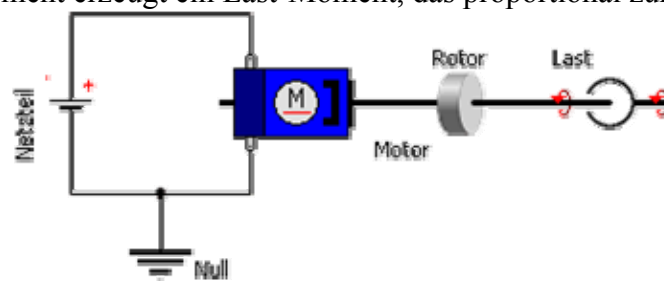


- Genauso lassen wir uns die Drehzahl in **U/min** anzeigen:

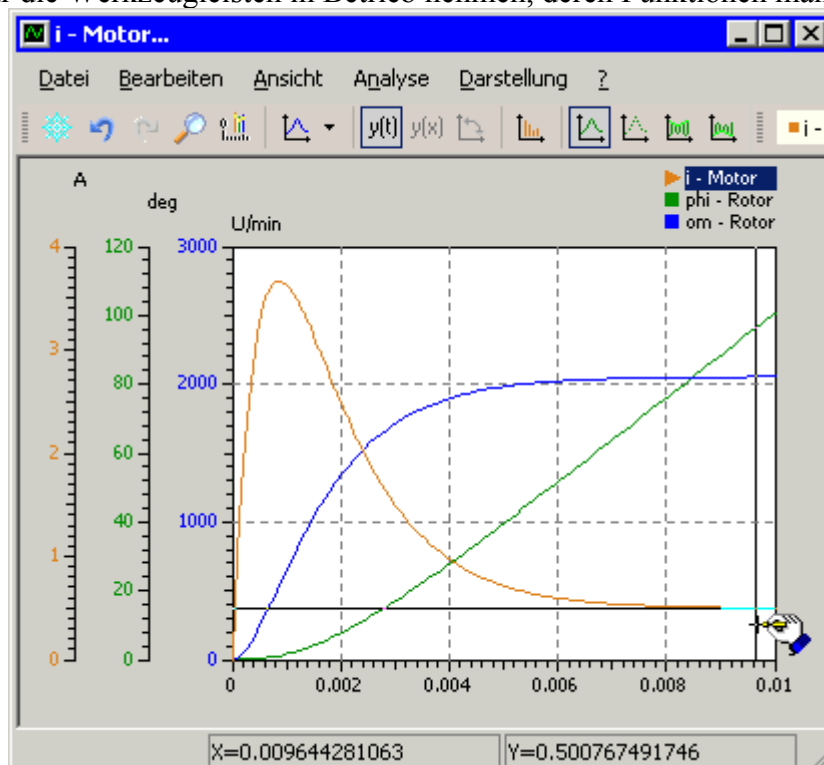


- Der verwendete Motor erreicht ohne Belastung eine Enddrehzahl von ungefähr 2300 U/min.
- Die Enddrehzahl ist erreicht, wenn der Motorstrom kein antreibendes Drehmoment mehr erzeugt. Dann wird der Rotor nicht mehr beschleunigt.
- Ein Motor wirkt gleichzeitig als Generator. Der Motorstrom geht im Leerlauf (ohne Reibung) auf Null, wenn die infolge der Rotor-Drehung induzierte Spulenspannung gleich der Betriebsspannung ist. Dann ist die Spannungsdifferenz über der Spule gleich Null.

Abschließend zu diesem Experiment werden wir den Rotor noch mit einer Dämpfung belasten (Rotatorische Mechanik). Ein Dämpfer-Element erzeugt ein Last-Moment, das proportional zur Drehzahl ist:



- Von Null beginnend wollen wir die Last (**Dämpfung b**) vorsichtig erhöhen, bis ein Endstrom von ungefähr  $i=0,5 \text{ A}$  fließt:
- Spätestes bei dieser iterativen Suche des erforderlichen Dämpfungswertes ist es sinnvoll, von der Menü-Benutzung auf die Werkzeugleiste **Simulation** umzusteigen. Diese sollte nun dauerhaft angezeigt und benutzt werden.
- **Hinweis:** Bei der weiteren Nutzung von SimulationX sollte man schrittweise nur die Werkzeugleisten in Betrieb nehmen, deren Funktionen man permanent benötigt!



- Die Enddrehzahl verringert sich bei dieser Belastung um ca. 10%.

- Die elektrische Leistung von  $12\text{ W} = 24\text{ V} \cdot 0,5\text{ A}$  wird teilweise in der Motor-Spule verheizt ( $1,25\text{ W} = 0,5^2\text{ A}^2 \cdot 5\text{ Ohm}$ ). Der überwiegende Teil von  $10,75\text{ W}$  entspricht der aufgewandten mechanischen Leistung.
- **Hinweis:** den Endstrom  $i(t_{\text{Stop}})$  kann man auf zwei Arten ermitteln
  - Wahl von  $i$ -Motor im Ergebnisfenster (z.B. über die Legende) und Wahl des Zeitpunktes X mit dem Cursor. Der Wert Y in der Fußzeile des Ergebnisfensters entspricht dann dem Stromwert. (Siehe vorheriges Bild).
  - Wahl des Elements Motor in der Modellansicht oder im Modellexplorer. Der aktuelle Wert des Motorstroms erscheint in der zugehörigen Liste der Ergebnisgrößen:

Kommentar	Name	Aktuell...	Maßeinheit
Ankerwiderstand	R	5	Ohm
Ankerinduktivität	L	2	mH
Drehmomentkonstante	kT	0.1	Nm/A
Spannungskonstante	kEMF	0.1	Vs/rad
Anfangsstrom	i0	0	A

Kommentar	Name	Aktueller Wert	Maßeinheit	Protokoll
Luftspaltmoment	T	0.0499375146449105	Nm	
Strom	i	0.499375146449105	A	
Spannung	v	24	V	

**Speichern des Modells nicht vergessen!**

← →

Von „[http://www.optiyummy.de/index.php/Software: SimX - Einfuehrung - DC-Motor - Drehzahl](http://www.optiyummy.de/index.php/Software:_SimX_-_Einfuehrung_-_DC-Motor_-_Drehzahl)“

# Software: SimX - Einfuehrung - DC-Motor - Regelkreis

Aus OptiYummy

↑

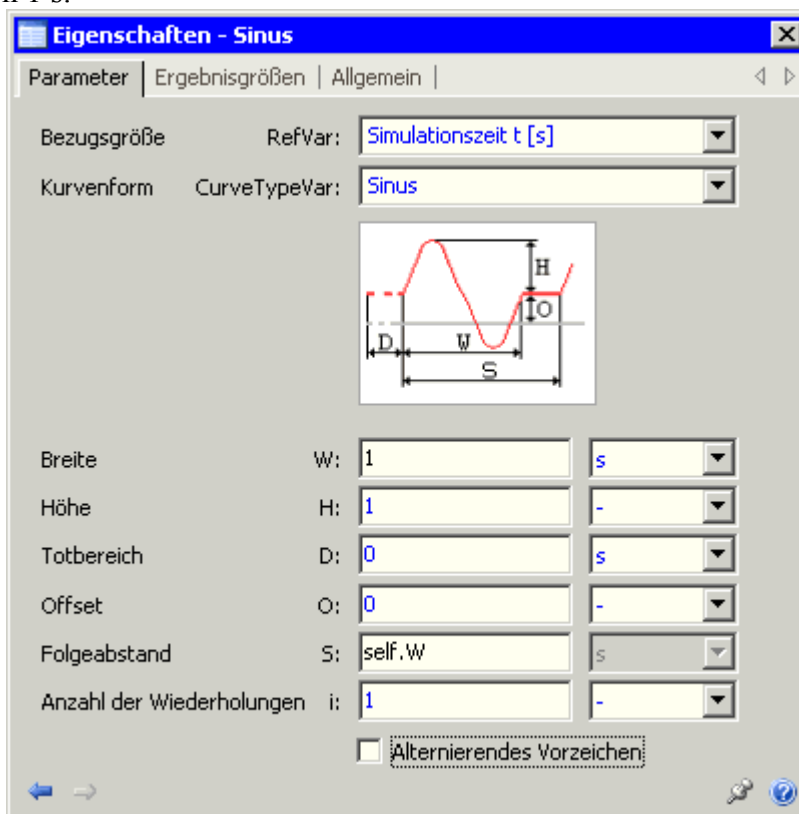
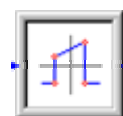
← →

## Regelkreis-Experiment

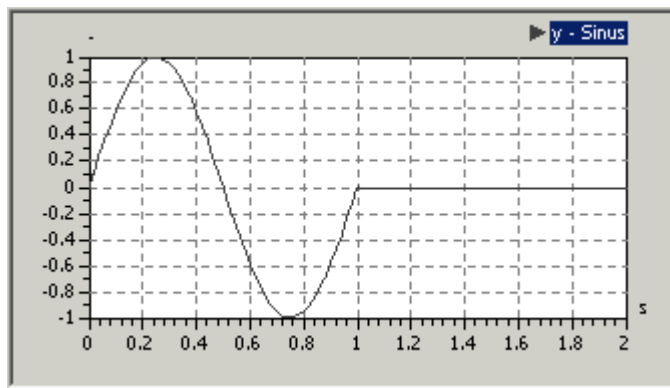
Oft soll ein Antrieb nicht nur ein- und ausgeschaltet werden, sondern sich nach einer vorgegebenen Sollkurve bewegen. Das kann man mit einem Gleichstrom-Motor hinreichend genau nur mit einem Regelkreis realisieren.

In unserem Beispiel soll der Motor erst in eine Richtung hochdrehen, dann die Bewegung umkehren. Zum Schluss soll er wieder stehen bleiben. Der gesamte Vorgang soll 1 Sekunde dauern. Bei der Bewegung soll der Motor möglichst ruckfrei beschleunigen, also ganz "zart" die Drehzahl ändern. Dafür wollen wir zuerst einen Sollwert-Generator aufbauen:

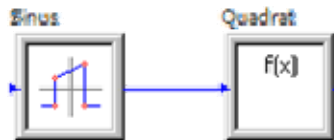
- Es bietet sich an, für diesen Bewegungsvorgang eine komplette Sinus-Schwingung (=1 Sinus-Impuls) als Grundlage zu nehmen.
- Wir finden in der Bibliothek das Element **Impulsgenerator** unter "Signalglieder - Signalquellen".
- Nach dem Einfügen in unser Antriebsmodell konfigurieren wir diesen Generator als Sinusgenerator mit einer Impulslänge von 1 s:



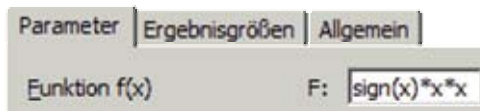
- **Hinweis:**  
Der Name *self* bezeichnet das eigene Element. SimulationX ergänzt diesen Vorsatz automatisch für alle Bezeichner, welche sich auf das eigene Element beziehen. Im Beispiel ist der Folgeabstand  $S$  also immer so groß wie die Impuls-Breite  $W$ .
- Wenn wir die Simulationszeit für das Modell auf 2 s erhöhen, erhalten wir folgendes Ausgangssignal vom Sinusgenerators:



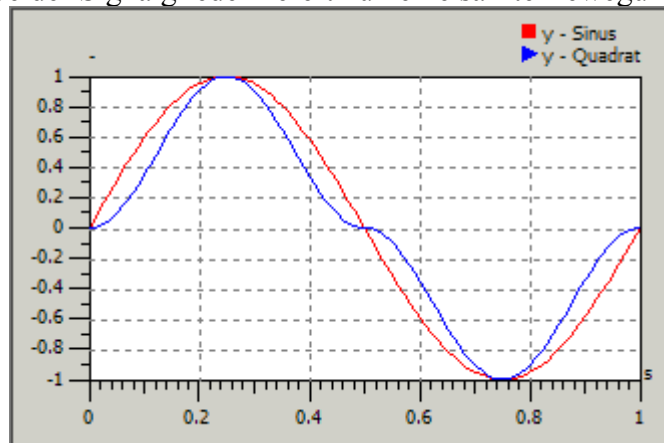
Ein solcher Drehzahl-Verlauf wäre aber noch nicht sanft genug. Am Anfang und Ende gäbe es einen ordentlichen Ruck! Deshalb quadrieren wir den Funktionsverlauf unter Beibehaltung des Vorzeichens:



- Wir nutzen dafür das Element  $f(x)$  direkt aus dem Bibliotheksordner "Signalglieder".
- $\text{sign}(x)$  liefert das Vorzeichen von  $x$  und kann nur 3 Werte besitzen (-1,0,+1):



- Die Zusammenschaltung beider Signalglieder liefert nun eine sanfte Bewegungssollkurve:



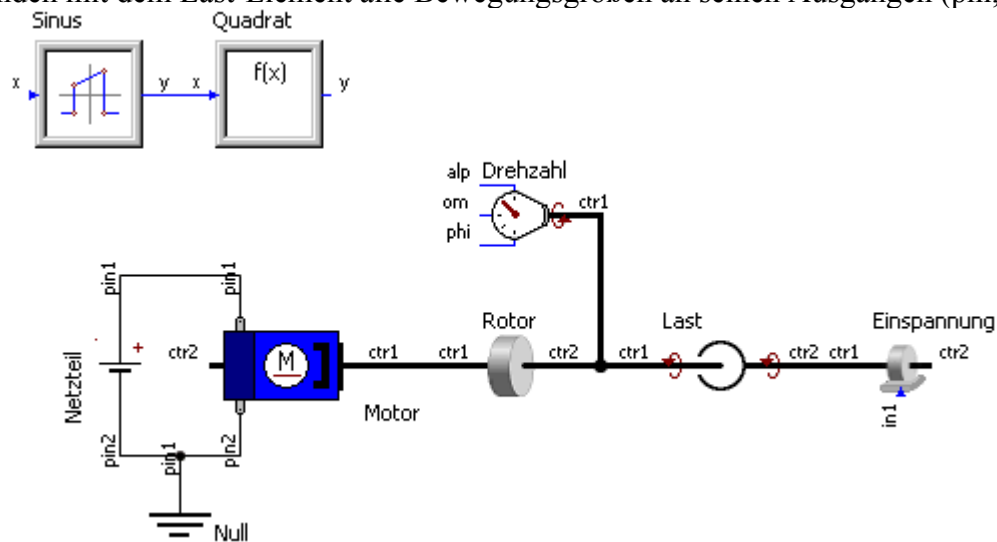
Zusätzlich zum Sollwert-Generator benötigen wir einen Sensor, welcher die aktuelle Drehzahl misst. In der Praxis funktioniert das z.B. mit einer Codescheibe in einem optischen inkrementalen Geber:



Wir wollen das im Beispiel stark vereinfacht mit einem trägheitslosen, analogen Sensor-Element aus der

Bibliothek realisieren:

- Den Sensor findet man in der Bibliothek im Ordner "Rotatorische Mechanik". Dieser Sensor liefert nach seinem Verbinden mit dem Last-Element alle Bewegungsgrößen an seinen Ausgängen (phi, om, alp):



**Toolbars "Element" und "Anzeige" einschalten:**

- Das Sensor-Element dreht man in der Modellstruktur in die gewünschte Position.
- Damit man sieht, an welchem Anschluss welches Signal anliegt, kann man die Anschlüsse innerhalb der Modellstruktur einblenden. Die Beschriftung sollte man dann wieder ausschalten, weil sie meist stört!

**Hinweise zum unverbundenen Anschlüssen:**

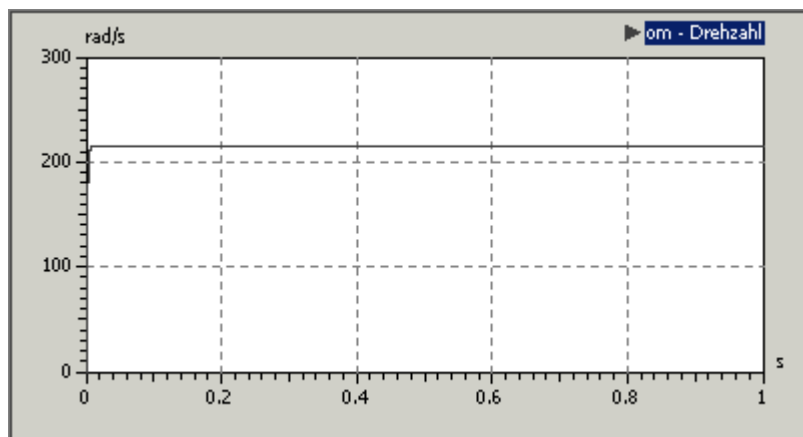
- Freie Anschlüsse eines Kraftelements (z.B. Feder, Dämpfer, Reibung) sind im Modell "automatisch" fest eingespannt. Um Fehler zu vermeiden, sollte man möglichst (wie gezeigt) die in der Realität vorhandene Einspannung als Element ergänzen (Rotatorische Mechanik - Element "Vorgabe" - konfiguriert als "Einspannung").
- Ein sehr "gemeiner" Modellierungsfehler wäre z.B. die Verbindung des Drehzahlsensors mit dem "freien" Anschluss der Last-Dämpfung. Dann würde die Dämpfung praktisch unwirksam, da nicht mehr die Relativbewegung zwischen dem Rotor und dem Motorgehäuse erfasst würde!
- Der "freie" Anschluss **Motor.ctr2** entspricht dem Motorgehäuse, das praktisch auch fest eingespannt ist. Wegen der Übersichtlichkeit wurde hier mit der Verbindung eines Einspann-Elements verzichtet.

Als Amplitude für unseren Sollwert-Verlauf wollen wir **100 rad/s** vorgeben. Dazu müssen wir die Quadrat-Funktion um diesen Faktor ergänzen):

$$F: \text{sign}(\text{self.x}) * \text{self.x} * \text{self.x} * 100$$

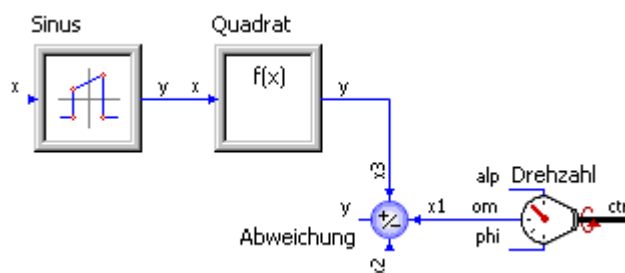
**Hinweise zu Maßeinheiten:**

- Die Modellberechnung in SimulationX erfolgt grundsätzlich mit den SI-Einheiten!
- Verwendet man eine Modellgröße in einer Formel, so enthält die Modellgröße immer den aktuellen Zahlenwert für die SI-Einheit.
- Deshalb wurde im Beispiel eine Drehzahlvorgabe in der SI-Einheit rad/s verwendet, ansonsten hätte man die Umrechnung in der Formel entsprechend berücksichtigen müssen!
- Nur für die Eingabe der Parameter und die Darstellung von Ergebnisgrößen können handliche Vorsätze zu den SI-Einheiten gewählt werden. SimulationX nimmt dann automatisch die Umrechnung zur SI-Einheit vor.
- Für die Darstellung des Sensor-Signals sollte im Beispiel ebenfalls die SI-Einheit benutzt werden:

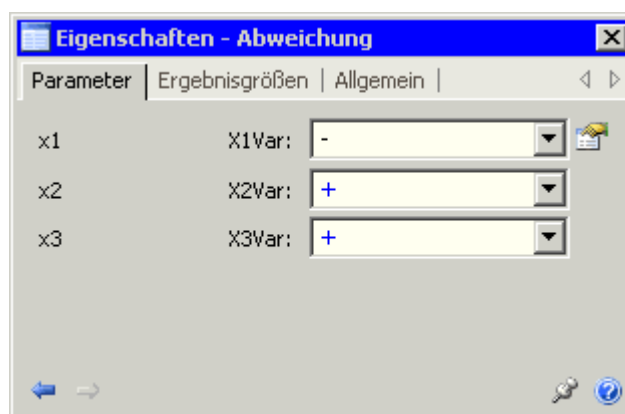


Im nächsten Schritt muss man die Regelabweichung als Differenz **Sollwert minus Istwert** bilden:

- Wir nutzen für die Bildung der Abweichung aus der Bibliothek einen Summationspunkt (direkt im Ordner "Signalglieder"):

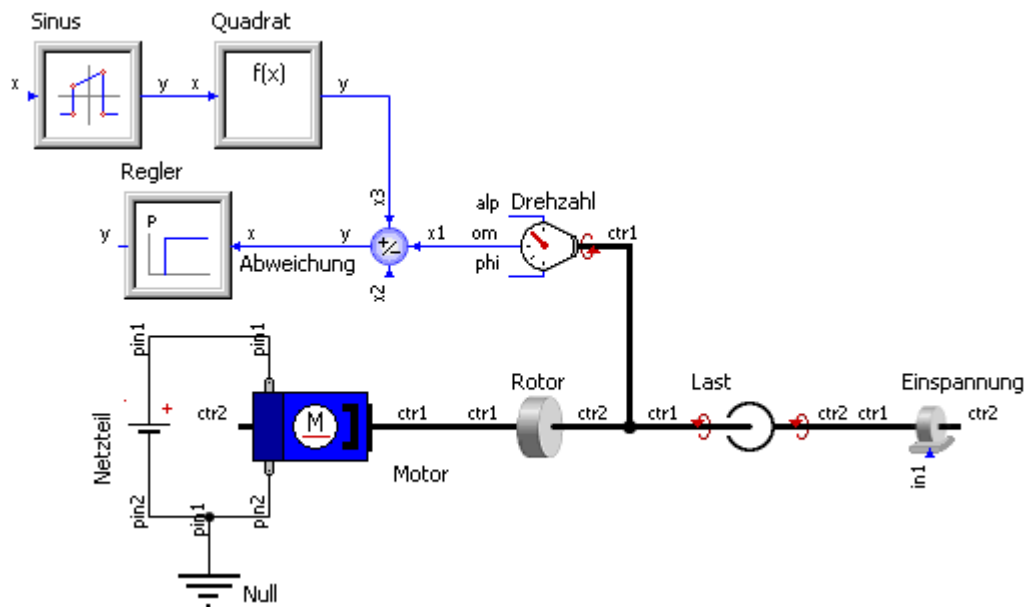


- Im Beispiel wurde dieses Element um 180° gedreht und der Eingang **x1** mit dem Istwert gespeist. Der Sollwert geht in den Eingang **x3**. Um  $Y=x3-x1$  zu bilden, muss man den Summationspunkt wie folgt konfigurieren:



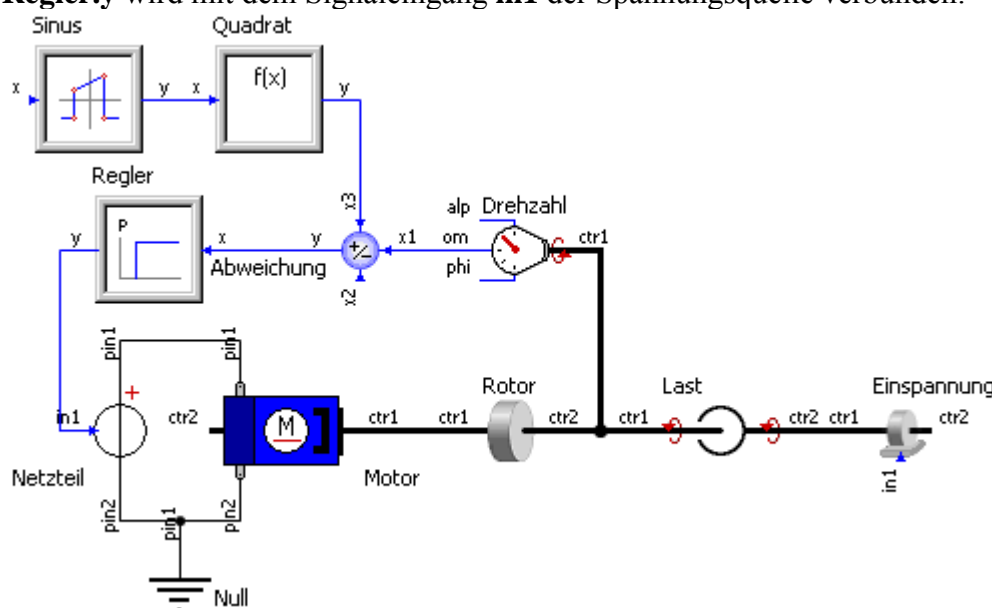
Der Regler wertet die Regel-Abweichung aus und beeinflusst dann die Betriebsspannung für den Motor so, dass diese Abweichung möglichst klein wird:

- Wir nutzen einen einfachen P-Regler (P-Element in der Bibliothek unter "Signalglieder - Lineare Signalglieder") und speisen seinen Eingang mit der Regelabweichung.
- Ein P-Regler verstärkt mit dem Faktor G die eingespeiste Regelabweichung (Proportional-Regler):

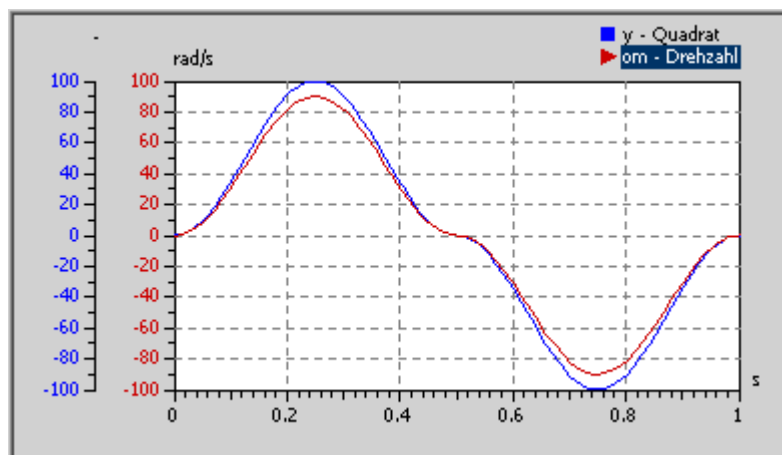


Mit dem resultierenden Ausgangswert des Reglers muss man nun entsprechend die Betriebsspannung für den Motor beeinflussen. Dazu benötigen wir eine gesteuerte Spannungsquelle:

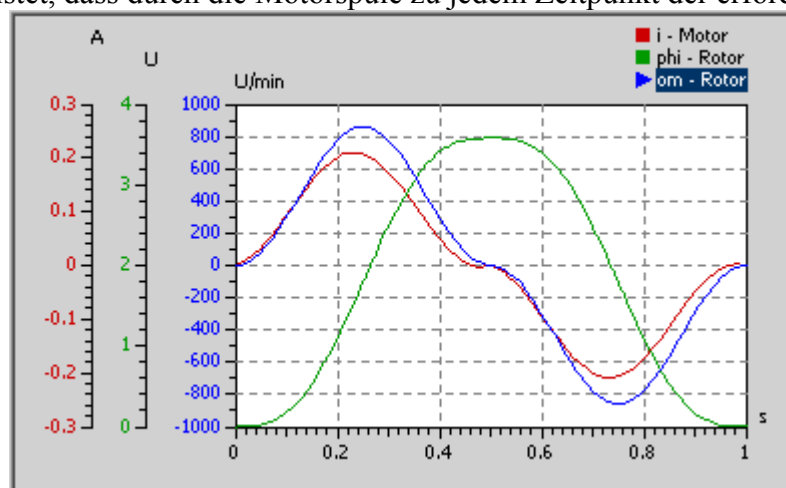
- Wir ersetzen die "Konstantspannungsquelle" (im Modell löschen) durch eine "Spannungsquelle mit Signaleingang" (unter "Elektronik - Analog - Quellen").
- Der Ausgang **Regler.y** wird mit dem Signaleingang **in1** der Spannungsquelle verbunden:



- Als Ausgangsspannung des Netzteils soll direkt der Ausgangswert **Regler.y** benutzt werden (entspricht **in1** des Netzteils).
- Durch Ändern des Verstärkungsfaktors **G** des Reglers kann man nun eine hinreichende Übereinstimmung zwischen Sollwert- und Istwert-Verlauf der Drehzahl erreichen.
- Es ist günstig, beide Signal-Verläufe in einem Fenster darzustellen.
- Die Verstärkung sollte man möglichst niedrig halten, weil hohe Verstärkungen zu instabilen Reglerverhalten führen.
- Der dargestellte Drehzahlverlauf ergibt sich bei  $G=1$ :



- Der Regler gewährleistet, dass durch die Motorspule zu jedem Zeitpunkt der erforderliche Strom fließt:



### Achtung!

Leider wird der Proportional-Regler in der Praxis nie so gut funktionieren wie hier im Modell:

- Im Modell wurden Zeitverzögerungen im Antriebsstrang, z.B. infolge von Nachgiebigkeiten in der Antriebswelle, nicht berücksichtigt.
- In der Praxis führt das zum Schwingen des Antriebssystems schon bei relativ kleinen Verstärkungen.
- Mit den kleinen Verstärkungen kann man nur eine ziemlich große Regelabweichung realisieren. Deshalb ergänzt man den Regler um einen Integral-Anteil. Damit kann man im Idealfall die Regelabweichung stark reduzieren.

### Speichern des Modells nicht vergessen!



Von „[http://www.optiyummy.de/index.php/Software: SimX - Einfuehrung - DC-Motor - Regelkreis](http://www.optiyummy.de/index.php/Software:_SimX_-_Einfuehrung_-_DC-Motor_-_Regelkreis)“

# Software: SimX - Einfuehrung - DC-Motor - Simulation

Aus OptiYummy

↑

← →

## Was ist nun eigentlich Simulation?



Umgangssprachlich ist ein Simulant einer, welcher nicht vorhandenen Symptome einer Krankheit vortäuscht, um daraus Vorteile zu ziehen. Diese Art Simulation wollen wir hier nicht betrachten! Bei uns geht es darum, durch Experimentieren neue Erkenntnisse zu gewinnen. Im betrachteten Beispiel wäre ein Erkenntnisziel, wie man den Regler dimensionieren sollte, damit der Antrieb sich wie gewünscht verhält.

Um diese Erkenntnis zu gewinnen, führten wir Experimente durch ("mal schauen, was passiert"). Allerdings verwendeten wir dabei nicht den echten Antrieb, sondern benutzten ein Computermodell.

### Verallgemeinert kann man sagen:

Simulation ist jegliche Benutzung eines Modells als Ersatzobjekt, um damit Erkenntnisse über das modellierte Objekt zu gewinnen.

### Dazu einige Erläuterungen:

- Simulation hat nicht unbedingt etwas mit der Benutzung eines Computers zu tun. Wenn man z.B. zwei Bälle und eine Kerze zur Verdeutlichung der Mondphasen benutzt, ist das auch eine Simulation.
- Auch wenn man über etwas nachdenkt, "simuliert" man. Man nutzt in den Gedankenexperimenten dann Sprach-Konstruktionen als Ersatz-Objekte für das, was in der Realität stattfinden soll.
- Simulation ist etwas typisch Menschliches. Denn bevor man eine geplante Handlung ausführt, hat man sich bereits anhand von Ersatz-Objekten (Modellen) überlegt, was dann passiert.
- Leider ist die "Simulationsleistung" des menschlichen Gehirns ziemlich beschränkt. Durch die Benutzung von Computer-Modellen kann der Mensch zu ganz neuen Erkenntnissen gelangen. Denn viele Operationen des menschlichen Nachdenkens kann man in Form von Computer-Programmen formulieren (im Sinne "if ... then ... else ...;"). So kann man mit Computer-Simulationen ganz neue Lösungen für komplizierte Problemstellungen finden!
- Bei aller Technik-Begeisterung sollte man jedoch beachten: "Nicht alles lässt sich simulieren!"



← →

Von „[http://www.optiyummy.de/index.php/Software: SimX - Einfuehrung - DC-Motor - Simulation](http://www.optiyummy.de/index.php/Software:_SimX_-_Einfuehrung_-_DC-Motor_-_Simulation)“