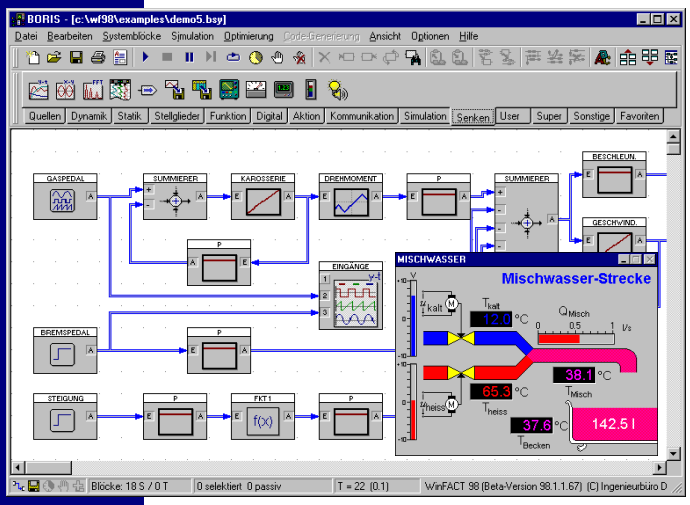
WinFACT 98



WinFACT 98

Benutzerhandbuch

Release 1.0

Ingenieurbüro Dr. Kahlert

Ludwig-Erhard-Str. 45
D-59065 Hamm

WinFACT 98

WINDOWS FUZZY AND CONTROL TOOLS

BENUTZERHANDBUCH

RELEASE 1.0

Ingenieurbüro Dr. Kahlert
Ludwig-Erhard-Str. 45 D-59065 Hamm
Tel. 0 23 81/926 996 Fax 0 23 81/926 997

Inhaltsverzeichnis

Willkommen

Willkommen zu WinFACT 98. Diese Version hat viele neue Merkmale, die Ihnen später im Detail erläutert werden. Dieses Kapitel gibt zunächst einige Hinweise, die Ihnen die Arbeit mit diesem Handbuch erleichtern sollen.

Damit Sie Informationen mühelos finden und identifizieren können, werden in diesem Handbuch optische Hilfen und Standard-Textformate verwendet. So finden Sie die folgenden typografischen Darstellungsweisen:

Zeichenformat	**Verwendet für**
KAPITÄLCHEN	Menüoptionen. Einzelne Menübefehle werden dabei durch einen senkrechten Strich voneinander getrennt. Beispiel: DATEI | ÖFFNEN... bezeichnet den Menüpunkt ÖFFNEN im Untermenü DATEI.
kursiv	1. Bezeichnungen von Kontrollelementen in Dialogfenstern (z. B. Editierfeldern, Auswahlfeldern etc.) 2. Neue Begriffe, die zum ersten Mal im Text erscheinen
fett	Benutzereingaben
GROSSBUCHSTABEN	Namen von Programmen oder Dateien
`Systemschrift`	Programmlistings oder Teile davon
<Shift><A>	Bezeichnungen von Tasten

Für das schnelle Auffinden bestimmter Informationen werden Kernbegriffe, wichtige Bedienungshinweise usw. zusätzlich durch Randbemerkungen (Marginalien) am linken Seitenrand herausgestellt. Auf bestimmte Arten von Informationen wird über besondere Symbole hingewiesen:

Symbol	Bedeutung
	Vertiefende Hinweise für interessierte Anwender, die für die Nutzung der Software nicht unbedingt erforderlich sind
	Weist auf Beispieldateien hin, die im Lieferumfang von WinFACT 98 enthalten sind
	Warnt vor "Stolpersteinen" bzw. Mißverständnissen, die bei speziellen Anwendungsfällen Probleme bereiten können
	Weist auf besondere Tips zum Umgang mit der Software hin
	Kennzeichnet wesentliche Neuheiten oder Erweiterungen von WinFACT 98 gegenüber WinFACT 96. Besonders interessant für erfahrene WinFACT 96-Anwender, die das Handbuch nach Neuheiten "überfliegen" wollen.

Auch die umfangreichsten Tests und die ausführlichsten Handbücher können keine hundertprozentige Sicherheit vor Softwareproblemen bieten. In solchen Fällen stehen wir Ihnen jederzeit telefonisch, über E-Mail oder über unsere Web-Site im Internet für Anfragen zur Verfügung[1]. Das gleiche gilt selbstverständlich auch für jegliche Art von (konstruktiver) Kritik, Lob oder Verbesserungsvorschläge. Ihr heißer Draht zu uns:

Telefon:	0 23 81/926-996
Fax:	0 23 81/926-997
E-Mail:	support@kahlert.com
Internet:	http://www.kahlert.com

[1] Bestimmte Dienstleistungen können ggf. einen Software-Pflegevertrag voraussetzen, der in Ergänzung zur Software abgeschlossen werden kann.

1 Erste Schritte

Installation und Konfigurierung von WinFACT 98

Lieferumfang

WinFACT 98 wird auf einer selbststartenden CD-ROM ausgeliefert, auf der sich alle zum Betrieb benötigten Dateien sowie weitergehende Informationen, Demo-Software usw. befinden. Zum Lieferumfang gehört darüber hinaus eine Komplettdokumentation (dieses Handbuch) sowie bei bestimmten Lizenzformen ein Hardware-Schutzstecker (Dongle) zum Anschluß an die parallele Schnittstelle des Rechners.

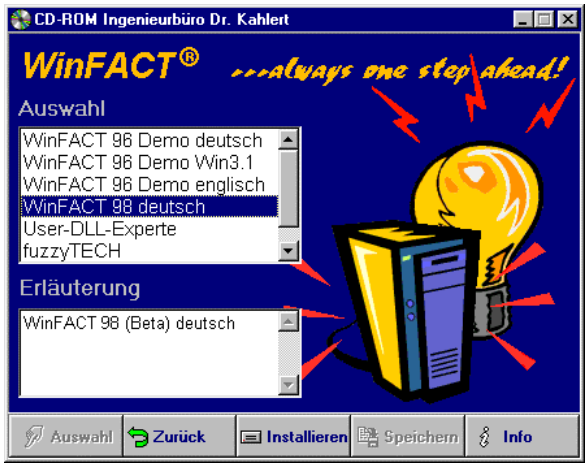
Installation

Installation von CD-ROM

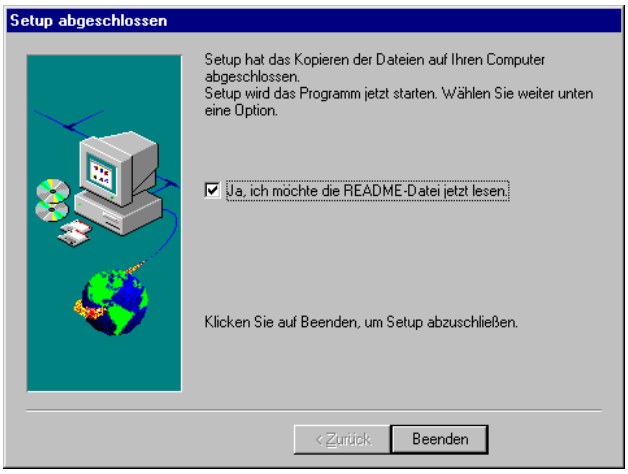
Nach dem Einlegen der CD-ROM in das CD-Laufwerk Ihres Rechners wird nach einigen Sekunden automatisch eine CD-Benutzeroberfläche gestartet[1], über die Sie auf einfache Weise WinFACT 98 installieren können. Dazu wählen Sie zunächst den Menüpunkt SOFTWARE und dann die zu installierende Version aus und starten die Installation über die *Installieren*-Schaltfläche.

Die Installation selbst läuft vollständig menügesteuert ab. Das abschließende Angebot, die aktuelle README-Datei zu lesen, sollten Sie unbedingt annehmen, um sich die letzten Informationen zur installierten Version anzeigen zu lassen!

[1] Sollte diese Option auf Ihrem Rechner deaktiviert sein, starten Sie einfach das Programm START.EXE aus dem Root-Verzeichnis der CD.



Benutzeroberfläche der WinFACT-CD



Lesen der README-Datei

Anfertigung eines Installations-Diskettensatzes

Bei Bedarf kann anhand der CD-Dateien ein Installations-Diskettensatz erstellt werden. Dazu kopieren Sie alle Dateien der Unterverzeichnisse *DISK1*, *DISK2*, *DISK3* usw. des Verzeichnisses *SOFTWARE\WF98* auf jeweils eine 1.44 MB-

Diskette. Zur Installation wird das Programm SETUP.EXE der ersten Diskette aufgerufen.

Installation des Hardware-Schutzsteckers

Bestimmte WinFACT 98-Lizenzen werden mit einem Hardware-Schutzstecker (Dongle) ausgeliefert, der auf die parallele Schnittstelle des Rechners montiert werden muß. Sofern Sie eine solche Lizenz erworben haben, muß vor dem ersten Aufruf eines WinFACT 98-Anwenderprogramms zunächst der Treiber für den Schutzstecker installiert werden. Dazu rufen Sie aus der WinFACT 98-Programmgruppe das Programm *Hardwareschutz installieren* (Datei DONGLESETUP.EXE) auf. Im daraufhin erscheinenden Dialog betätigen Sie nach Wahl des Betriebssystems die Schaltfläche *Start*; alle notwendigen Treiber werden dann automatisch installiert. Zum Abschluß muß der Rechner einmalig neu gebootet werden, bevor erstmalig ein WinFACT 98-Anwendungsprogramm aufgerufen werden kann.

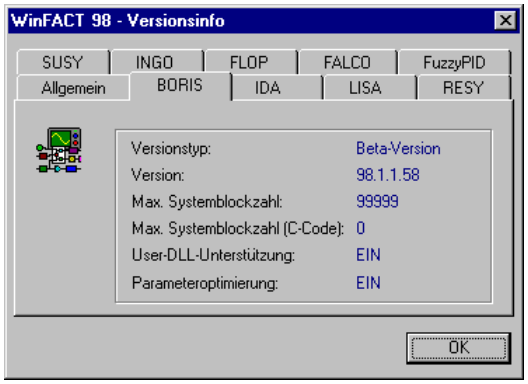
Installation der Hardware-Schutzstecker-Treiber

Einschränkungen bestimmter Lizenzformen

WinFACT 98 ist ein modular aufgebautes Programmsystem, das in einer Vielzahl unterschiedlicher Versionen und Lizenzformen erworben werden kann. Welche der in diesem Handbuch beschriebenen WinFACT 98-Komponenten in dem von Ihnen erworbenen Paket tatsächlich vorhanden sind und installiert werden, hängt daher vom Typ der erworbenen Version bzw. Lizenz ab. Je nach Typ können einige der Komponenten eingeschränkt sein (z. B. in Studenten- oder Ausbildungslizenzen) oder auch ganz fehlen bzw. nur als Demo-Version ausgelegt sein.

Abfrage von Versionsinformationen

Für Supportzwecke ist es wichtig, jederzeit die Versionsnummern der auf Ihrem Rechner installierten WinFACT 98-Anwendungsprogramme zu kennen. Diese können einerseits aus dem *Info über...*-Dialog des jeweiligen Anwendungsprogramms erfragt werden, sind aber auch zentral verfügbar. Dazu dient das Programm *WinFACT 98-Versionsinfo* (Datei WFINFO.EXE) aus der WinFACT 98-Programmgruppe. Es liefert für sämtliche WinFACT 98-Programmodule alle relevanten Versionsinformationen.



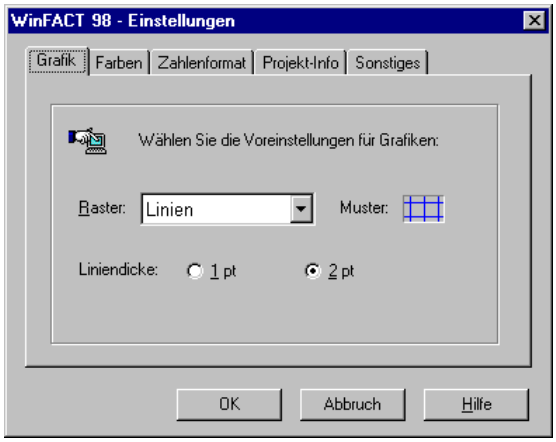
WinFACT 98-Versionsinfo (hier für Modul BORIS*)*

Konfigurierung von WinFACT 98

Die Konfigurierung von WinFACT 98 umfaßt die Voreinstellung bestimmter Parameter, die sämtliche oder zumindest die meisten WinFACT 98-Einzelkomponenten betreffen. Zur Konfigurierung starten Sie das Programm *WinFACT 98-Einstellungen* (Datei WFSETUP.EXE) aus der WinFACT 98-Programmgruppe. Die verschiedenen Optionen sind auf fünf Palettenseiten untergebracht, die sich über die Maus anwählen lassen. Alle im Konfigurie-

rungsprogramm getätigten Voreinstellungen werden in der Windows-Registrierung abgespeichert.

Palettenseite *Grafik*



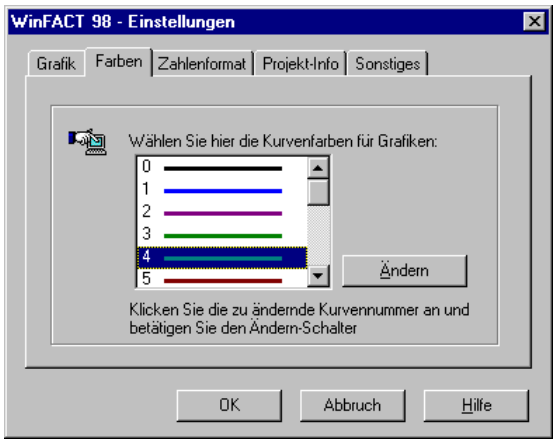
Palettenseite Grafik

Diese Palettenseite enthält folgende Grafik-Optionen:

Raster Über das Kombinationsfenster *Raster* läßt sich der voreingestellte Rastertyp wählen. Diese Voreinstellung kann in einigen Programmen (z. B. INGO) nachträglich geändert werden.

Liniendicke Über *Liniendicke* läßt sich die Standardbreite von Kurven festlegen. Auch diese Einstellung kann in einigen Modulen geändert werden. Man beachte in diesem Zusammenhang, daß die Liniendicke beim Ausdrucken von Grafiken in der Regel automatisch angepaßt wird.

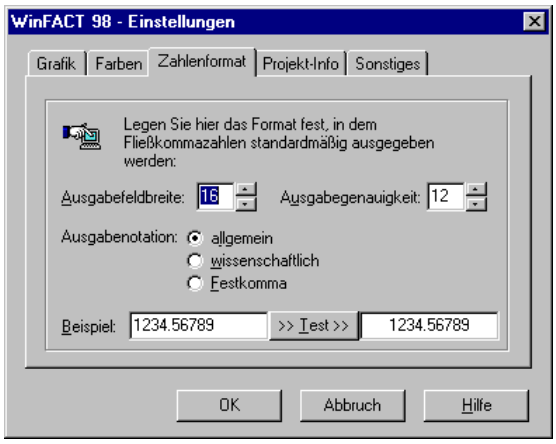
Palettenseite *Farben*



Palettenseite Farben

Über diese Palettenseite lassen sich die Kurvenfarben wählen, mit denen innerhalb des Moduls INGO mehrere Kurven in ein Diagramm gezeichnet werden. Beachten Sie dabei, daß die Farben innerhalb dieser Module erst bei Kurvennummer 1 (nicht 0) beginnen. Farbe 0 (per Voreinstellung schwarz) sollte daher innerhalb des Setup-Programms nicht geändert werden. Zum Ändern einer Farbe klicken Sie mit der linken Maustaste zunächst die zu ändernde Farbnummer an und betätigen dann den *Ändern*-Schalter. Es erscheint daraufhin der Windows-Standarddialog zur Farbauswahl.

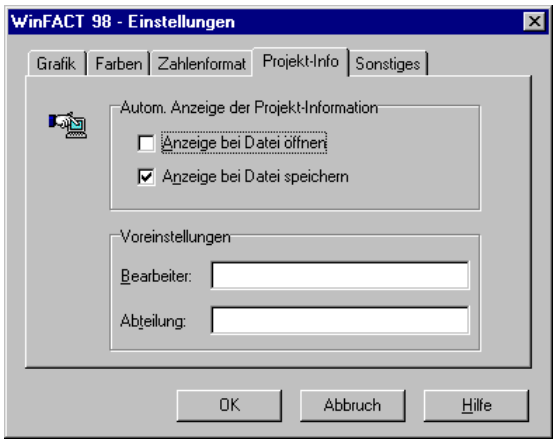
Palettenseite *Zahlenformat*



Palettenseite Zahlenformat

Über diese Palettenseite läßt sich das Standard-Ausgabeformat für Fließkommazahlen (z. B. beim Abspeichern in Datei) festlegen. Die aktuell gewählte Einstellung kann durch Eingabe einer beliebigen Fließkommazahl in das *Beispiel*-Eingabefeld und Betätigung der >> *Test* >>-Schaltfläche überprüft werden.

Palettenseite *Projekt-Info*



Palettenseite Projekt-Info

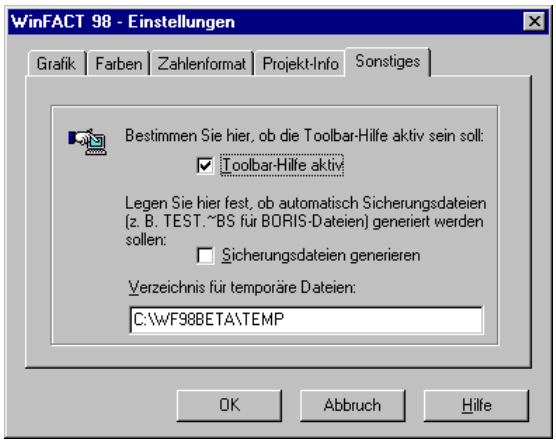
Die meisten WinFACT 98-Dateitypen können vom Anwender mit einer Projekt-Information versehen werden, um einen schnellen Überblick über ihren Inhalt zu erhalten. Dieses Registerblatt erlaubt dazu einige Voreinstellungen:

Anzeige bei Datei öffnen — Ist diese Option aktiviert, wird beim Öffnen einer Datei automatisch die entsprechende Projekt-Information angezeigt.

Anzeige bei Datei speichern — Ist diese Option aktiviert, wird beim erstmaligen Speichern einer Datei automatisch die entsprechende Projekt-Information abgefragt.

Voreinstellungen — Im Gruppenfeld *Voreinstellungen* können die Projekt-Info-Angaben für *Bearbeiter* und *Abteilung* voreingestellt werden, so daß ihre Eingabe bei der Abfrage der Projekt-Information jeweils entfallen kann.

Palettenseite *Sonstiges*



Palettenseite Sonstiges

Diese Seite ermöglicht folgende Voreinstellungen:

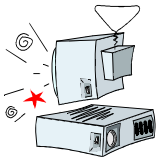
Toolbar-Hilfe aktiv	Ist diese Option aktiv, erscheint zu jeder Toolbar-Schaltfläche ein Hilfefenster (*Tool-Tip*), sobald sich der Mauscursor länger als etwa eine Sekunde darüber befindet.
Sicherungsdateien generieren	Ist diese Option aktiv, so werden zu den meisten WinFACT 98-Systemdateien vor dem Überschreiben mit neuem Inhalt Sicherungsdateien angelegt. Diese tragen denselben Namen wie die ursprüngliche Datei, wobei die Dateierweiterung jedoch eine führende Tilde (~) enthält. **Beispiel**: Die Sicherungsdatei zur BORIS-Datei TEST.BSY heißt TEST.~BS
Verzeichnis für temporäre Dateien	Legt das Verzeichnis für temporäre Dateien fest

2 Grundlagen

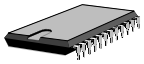
Technische Voraussetzungen

Was Ihr Rechner leisten sollte

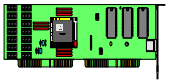
Eine Arbeit mit WinFACT 98 ist bereits auf Standard-Hardware möglich. Wie immer gilt auch hier, daß Arbeitskomfort und -geschwindigkeit mit der Leistungsfähigkeit der Rechnerhardware steigen. Die Mindestanforderungen sind:



ein Rechner vom Typ 486, Pentium oder dazu kompatibel (sinnvoll: Pentium oder höher)



mindestens 8 MB Hauptspeicher (sinnvoll: mind. 32 MB)



ein Grafikadapter mit einer Mindestauflösung von 800x600 Punkten bei 256 Farben

freier Festplattenspeicher von mindestens 20 MB

eine Maus, ein Trackball oder ähnliches

die grafische Benutzeroberfläche Windows 95, Windows 98 oder Windows NT

Einführung und Programmphilosophie

Was ist WinFACT 98?

WinFACT 98 stellt ein modular aufgebautes Programmsystem dar, das einerseits Werkzeuge zur Analyse, Synthese und Simulation konventioneller Regelungssysteme zur Verfügung stellt, andererseits aber insbesondere auch Komponenten zur Behandlung von Fuzzy-Systemen beinhaltet. Durch die grafische Benutzeroberfläche Windows wird ein extrem geringer Einarbeitungsaufwand bei gleichzeitig hohem Bedienungskomfort gewährleistet. Eine Vielzahl von Programmschnittstellen und Datenformaten ermöglicht die Kommunikation mit den unterschiedlichsten Peripheriegeräten, externen Prozessen und anwendereigenen Softwareprodukten sowie Fremdsoftware. Aus diesem Grunde ist WinFACT 98 sowohl für die Lehre als auch für Forschung und Entwicklung von Interesse.

WinFACT 98 besteht aus einer Zusammenstellung einzelner, im Prinzip unabhängiger und beliebig kombinierbarer Programmodule, zwischen denen auf äußerst einfache Weise ein Datentransfer über verschiedene Kommunikationskanäle möglich ist. Das Programmsystem in der Komplettversion enthält zunächst alle Komponenten, die zur Analyse und Synthese konventioneller Regelkreise erforderlich sind. Dazu gehören:

- Die Identifikation linearer Systeme anhand gemessener Verläufe der Ein- und Ausgangsgröße. Der eingesetzte Identifikationsalgorithmus zeichnet sich insbesondere durch seine Robustheit gegenüber Störungen wie z. B. Meßrauschen aus und ist bei beliebigen Typen von Eingangssignalen anwendbar.
- Die Analyse linearer Übertragungssysteme durch Berechnung von Sprungantwort, Bode-Diagramm, Ortskurve, Wurzelortskurve sowie Pol-Nullstellenverteilung.
- Die Synthese linearer Regler. Als Reglerkomponenten stehen alle gebräuchlichen Standardglieder zur Verfügung.
- Die Simulation und Optimierung von Regelkreisen

Fuzzy-Logik und Fuzzy Control

Neben konventionellen Methoden liegt ein Schwerpunkt des Programmkonzeptes im Bereich neuartiger Verfahren wie *Fuzzy-Logik* und *Fuzzy Control*. WinFACT 98 bietet hierzu Module an, die sämtliche Ebenen beginnend bei der Durchführung von "Fuzzy-Logik-Experimenten" über den interaktiven Entwurf regelbasierter Systeme bis hin zur Synthese und Simulation komplexer Fuzzy-Regelkreise erschließen. Dabei können alle Freiheitsgrade, die die Fuzzy-Logik bietet, vollständig ausgeschöpft werden:

- Unterschiedliche Typen von Fuzzy-Sets (dreiecksförmig, trapezförmig, Singleton)
- Verschiedene Verknüpfungsoperatoren und Inferenzmechanismen
- Vielzahl von Defuzzifizierungsmethoden

Entwurf und Analyse hybrider Systeme - bestehend aus konventionellen und Fuzzy-Komponenten - ist auf der Basis einer blockorientierten Simulation möglich. Komponenten beispielsweise zur grafischen Auswertung von Meß- oder Simulationsergebnissen auf Basis des Windows-MDI-Standards vervollständigen das Programmsystem.

Das Programmsystem weist die Windows-typische Benutzeroberfläche auf. Dies äußert sich für den Anwender speziell in der charakteristischen Menüstruktur mit der optionalen Anwahl über Tastenkürzel oder Toolbars, der integrierten Hilfefunktion und den gewohnt komfortablen Eingabedialogen, die Fehleingaben weitestgehend ausschließen.

Hardware-Schnittstellen

WinFACT 98 besitzt eine ganze Reihe verschiedenartiger Hardware-Schnittstellen, über die eine Prozeßankopplung (beispielsweise zur Meßdatenerfassung oder Regelung) möglich ist oder sich mit WinFACT 98 entworfene Strukturen (z. B. Regler) auf die Zielhardware portieren lassen:

- Über *A/D-D/A-Karten*. Nahezu alle handelsüblichen Kartentypen werden von WinFACT 98 unterstützt (z. B. Advantech, Meilhaus, National Instruments, Keithley, Sorcus, Wasco, Bitzer, Leybold, Analog Devices, Intelligent Instrumentation, PCMCIA-Cards usw.). Der Einsatz dieser Karten empfiehlt sich insbesondere für Ausbildungszwecke und Prototypenregelungen am realen Prozeß.
- Über die *serielle Schnittstelle* in Verbindung mit entsprechenden Hardwaresystemen (z. B. ISM-Serie der Fa. Gantner, ADAM-Module der

Fa. Advantech). Diese Realisierungsform ist besonders geeignet für Meßwerterfassung und zeitunkritische Anwendungen.

- Für die PC-unabhängige, autarke Realisierung bietet sich der äußerst leistungsfähige *C-Code-Generator* an. Er "übersetzt" beliebige Systemstrukturen in universellen ANSI-C-Code, der dann auf praktisch jede Zielhardware (z. B. Microcontrollerboards mit Standard-Controllern wie 68HC11, 80C166 usw. oder DSPs, SPS, ...) übertragen werden kann. Auf diese Weise sind auch hochdynamische Prozesse in Echtzeit steuer- und regelbar.

Einzelheiten sind den entsprechenden Produktinformationen zu entnehmen.

WinFACT-Dateiformate

Die in WinFACT benutzten Dateien zur Speicherung systemspezifischer Daten sind grundsätzlich ASCII-Dateien. Dies hat für den Anwender den Vorteil, daß er über die Verarbeitung in WinFACT hinaus die Daten jederzeit mit Hilfe eines gewöhnlichen Texteditors einsehen und gegebenenfalls auch modifizieren kann. Da die auftretenden Dateien in der Regel einige hundert Zeilen nicht überschreiten, hat die Wahl von ASCII-Dateien auf die Ein- und Ausgabegeschwindigkeit nur einen unwesentlichen Einfluß.

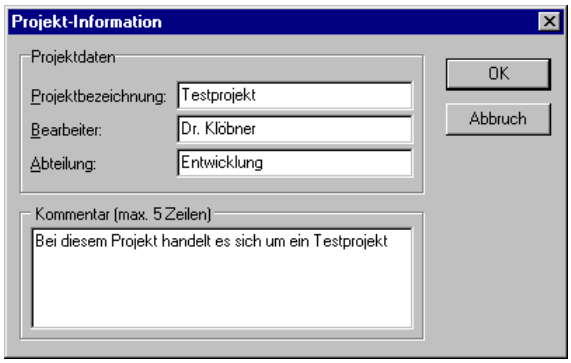
Für die unterschiedlichen Typen der innerhalb von WinFACT verwendeten Daten werden Dateien mit unterschiedlichen Erweiterungen (*Extensions*) benutzt, deren Format im folgenden beschrieben wird.

Projekt-Information

Alle Dateitypen können zur näheren Beschreibung des Dateiinhalts eine *Projekt-Information* enthalten, die bis zu acht Zeilen umfassen kann und am Dateianfang vor den eigentlichen Daten angeordnet ist. Die entsprechenden Dateizeilen sind durch ein führendes Ausrufezeichen gekennzeichnet. Die ersten fünf dieser Zeilen sind für die *Projektbeschreibung* reserviert. Zeile sechs kann einen *Projekttitel* enthalten, Zeile sieben den Namen des *Projektbearbeiters*

und Zeile acht seine *Abteilung*. Die Projektinformation wird beim Einlesen des Datensatzes - sofern gewünscht - vom jeweiligen Einzelprogramm angezeigt und beim erstmaligen Speichern einer Datei auf Wunsch abgefragt (siehe dazu den Abschnitt *Konfigurierung von WinFACT 98* im Kapitel *Erste Schritte* weiter vorn in diesem Handbuch).

Nachfolgende Bilder zeigen den Projekt-Info-Dialog und den zu den eingegebenen Daten generierten Dateikopf. Die entsprechenden Projekt-Informationen werden im DATEI ÖFFNEN-Dialog von WinFACT 98 bereits vor dem eigentlichen Öffnen einer Datei angezeigt, so daß ein Auffinden einer gesuchten Datei anhand der Projekt-Information sehr einfach möglich ist.



Dialog zur Anzeige und Modifikation des Projekt-Infos

```
! Hier können Sie
! bis zu 5 Zeilen mit
! Anmerkungen zu
! Ihrem Projekt
! unterbringen!!
! Demo-Projekt
! Müller-Lüdenscheid
! Software-Entwicklung
```

Zu obigem Projekt-Info-Dialog generierter Dateikopf

Dateitypen

Übertragungsfunktionen (UFK-Dateien)

Übertragungsfunktionen linearer Systeme mit Totzeit der Form

$$G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0} \; e^{-Ts}$$

werden in Dateien mit der Extension UFK und folgender Struktur abgelegt:[2]

```
m
n
b0
.
.
bm
a0
.
.
an
T
```

Simulationsergebnisse (SIM-Dateien)

Eine Auflistung von Wertepaaren (t_i, y_i), wie sie beispielsweise in Form von Simulationsergebnissen entstehen können, werden von WinFACT paarweise untereinander und durch mindestens ein Leerzeichen getrennt dargestellt. Die Anzahl der Wertepaare wird automatisch ermittelt. Die Dateien haben die Extension SIM.

```
t1   y1
t2   y2
t3   y3
.    .
.    .
```

WinFACT-Datei mit Wertepaaren

[2] Die Projekt-Information wird bei den nachfolgenden Dateilistings der Übersichtlichkeit halber weggelassen

Allgemeine Wertepaare (XY-Dateien)

Zur Speicherung allgemeiner Wertepaare (x_i, y_i) werden Dateien mit der Extension XY benutzt. Das Dateiformat ist mit dem von SIM-Dateien identisch.

Mehrfachwertepaare (MXY-Dateien)

MXY-Dateien eignen sich zur Speicherung mehrerer Kurvenzüge (z. B. Simulationsergebnisse) in einer einzigen Datei und werden in WinFACT zur Speicherung von Trajektorienfeldern im Modul SUSY benutzt. Jeder Kurvenzug enthält zu Beginn die Anzahl seiner Werte und danach die Wertepaare im Format wie bei SIM- bzw. XY-Dateien. Nach dem ersten Kurvenzug können beliebig viele weitere Kurvenzüge folgen. Das Ende der Datei wird automatisch erkannt, so daß die Angabe der Anzahl der enthaltenen Kurvenzüge nicht erforderlich ist.

Frequenzgänge (BD- bzw. OK-Dateien)

Der Frequenzgang $G(\mathrm{j}\omega)$ eines linearen Übertragungssystems kann in zweierlei Form dargestellt werden:

- Als *Bode-Diagramm* in Form von Wertetripeln

$$\left(\omega_i, \left|G(\mathrm{j}\omega_i)\right|_{\mathrm{dB}}, \angle G(\mathrm{j}\omega_i)\right)$$

  bestehend aus Frequenz, Betrag in dB und Phase in Grad. Die zugehörige Datei weist die Extension BD und die folgende Struktur auf:

ω_1	$\left\|G(\mathrm{j}\omega_1)\right\|_{\mathrm{dB}}$	$\angle G(\mathrm{j}\omega_1)$
ω_2	$\left\|G(\mathrm{j}\omega_2)\right\|_{\mathrm{dB}}$	$\angle G(\mathrm{j}\omega_2)$
ω_3	$\left\|G(\mathrm{j}\omega_3)\right\|_{\mathrm{dB}}$	$\angle G(\mathrm{j}\omega_3)$
.	.	.
.	.	.
.	.	.

- Als *Ortskurve* in Form von Wertetripeln

$$\left(\omega_i,\ \mathrm{Re}\{G(\mathrm{j}\omega_i)\},\ \mathrm{Im}\{G(\mathrm{j}\omega_i)\}\right)$$

bestehend aus Frequenz, Real- und Imaginärteil. Die Datei weist in diesem Fall die Extension OK und die folgende Struktur auf:

ω_1	$\mathrm{Re}\{G(\mathrm{j}\omega_1)\}$	$\mathrm{Im}\{G(\mathrm{j}\omega_1)\}$
ω_2	$\mathrm{Re}\{G(\mathrm{j}\omega_2)\}$	$\mathrm{Im}\{G(\mathrm{j}\omega_2)\}$
ω_3	$\mathrm{Re}\{G(\mathrm{j}\omega_3)\}$	$\mathrm{Im}\{G(\mathrm{j}\omega_3)\}$
.	.	.
.	.	.
.	.	.

Vektoren (VEK-Dateien)

Vektoren werden komponentenweise in Dateien mit der Extension VEK abgespeichert; jede Zeile der Datei enthält eine Komponente. Die erste Zeile nach der Projekt-Information enthält die Anzahl der Vektorkomponenten. Der Vektor

$$\underline{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

führt also zu folgender Datei:

```
n
b1
b2
.
.
bn
```

Komplexe Vektoren (KVK-Dateien)

Komplexe Vektoren (z. B. Eigenwertvektoren) der Form

$$\underline{r} = \begin{pmatrix} a_1 + jb_1 \\ a_2 + jb_2 \\ \vdots \\ a_n + jb_n \end{pmatrix}$$

werden in Dateien mit der Extension KVK wie folgt abgelegt:

```
n
a1  b1
a2  b2
.
.
an  bn
```

WinFACT-Datei mit komplexem Vektor

Matrizen (MAT-Dateien)

Matrizen werden zeilenweise abgespeichert, wobei jede Zeile genau ein Element enthält. Die erste Dateizeile nach der Projekt-Information enthält die Zeilen- und Spaltenzahl, durch ein Leerzeichen getrennt. Matrizendateien erhalten die Extension MAT. Eine $m \times n$ -Matrix

$$\underline{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

wird also in folgender Struktur abgelegt:

```
m n
a11
a12
.
.
a1n
a21
a22
.
.
```

Zustandsraummodelle (ZRM-Dateien)

Zustandsraummodelle der Form

$$\underline{\dot{x}} = \underline{\underline{A}}\,\underline{x} + \underline{b}\,u, \qquad y = \underline{c}^{\mathrm{T}}\,\underline{x} + d\,u$$

mit der Systemmatrix

$$\underline{\underline{A}} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

dem Steuervektor

$$\underline{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

und dem Ausgangsvektor

$$\underline{c}^{\mathrm{T}} = (c_1, c_2, \ldots, c_n)$$

werden in Dateien mit der Extension ZRM gemäß folgender Struktur abgelegt:

```
n
a11 a12 ... a1n
a21 a22 ... a2n
 .   .  ... .
 .   .  ... .
an1 an2 ... ann
b1
b2
.
.
bn
c1
c2
.
.
cn
d
```

Funktionswertmatrizen (FWM-Dateien)

Funktionswertmatrizen enthalten eine Funktion zweier Veränderlicher

$$z = f(x, y)$$

in diskretisierter Form, d. h. in Form diskreter Stützstellen

$$z_{ij} = f(x_i, y_j),\ i = 1,\ldots,m \quad j = 1,\ldots,n$$

auf einem Wertebereich $x_{\min} \le x \le x_{\max},\ y_{\min} \le y \le y_{\max}$. Diese Dateien lassen sich zur Darstellung der Funktion als 3D-Kennfeld oder in Höhenlinienform nutzen, beispielsweise mit Hilfe des WinFACT - Moduls INGO oder Programmen wie Mathematica.

Beispiel: Für die Funktion

$$z = x^2 + y^2, \qquad 0 \le x \le 5,\ 0 \le y \le 10$$

ergibt sich für m = 6 Stützstellen für x und n = 6 Stützstellen für y folgende Funktionswertmatrix:

$$\underline{A} = \begin{pmatrix} 0 & 4 & 16 & 36 & 64 & 100 \\ 1 & 5 & 17 & 37 & 65 & 101 \\ 4 & 8 & 20 & 40 & 68 & 104 \\ 9 & 13 & 25 & 45 & 73 & 109 \\ 16 & 20 & 32 & 52 & 80 & 116 \\ 25 & 29 & 41 & 61 & 89 & 125 \end{pmatrix}$$

Das Dateiformat entspricht dem von MAT-Dateien, die Datei hat also folgende Struktur:

```
m n
a11
a12
.
.
a1n
a21
a22
.
.
```

Spezielle Hilfefunktionen

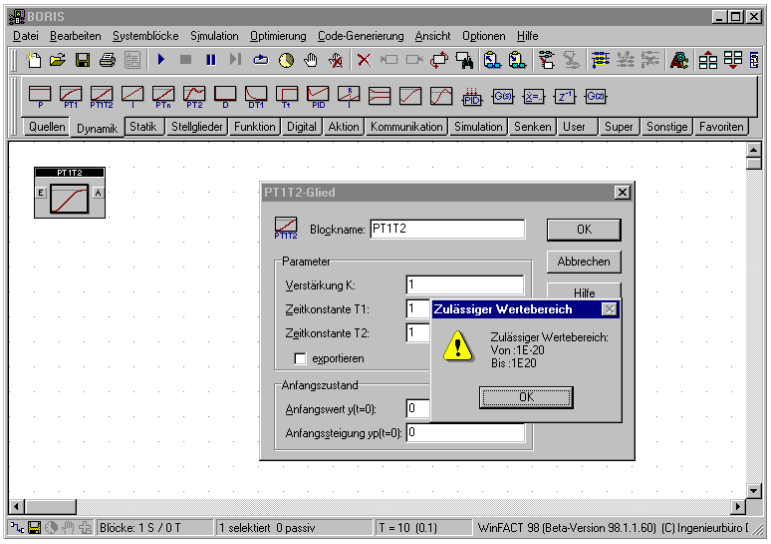
Toolbar-Hilfe

Eine wesentliche Erleichterung insbesondere für den Neueinsteiger stellt die über das Setup-Programm WFSETUP aktivierbare Toolbar-Hilfe dar. Ist sie aktiviert, erscheint automatisch ein Hilfefenster mit einer Erläuterung der Schaltflächen-Funktion, sobald sich der Cursor länger als etwa eine Sekunde über der Schaltfläche befindet. So lassen sich die mit den verschiedenen Schaltflächen verbundenen Funktionen auf einfache Weise erlernen.

Toolbar-Hilfefunktion

Wertebereiche numerischer Parameter

Bei der Eingabe numerischer Werte über einen entsprechenden Eingabedialog ist es häufig wünschenswert, den zulässigen Wertebereich vorab zu kennen, um von entsprechenden Fehlermeldungen und Warnungen "verschont" zu bleiben. Daher wurde in WinFACT 98 eine automatische Bereichsangabe integriert: Bei numerischen Eingabefeldern in Parameterdialogen genügt ein Anklicken mit der *rechten* Maustaste, um ein Meldungsfenster mit der Angabe des zulässigen Wertebereichs erscheinen zu lassen.



Anzeige des zulässigen Wertebereichs für numerische Parameter

3 Systemidentifikation mit IDA

Leistungsumfang

IDA ermöglicht die Identifikation linearer Systeme anhand des gemessenen Ein-/Ausgangsverhaltens. Die Eingangsgröße kann prinzipiell beliebigen Verlauf aufweisen. Zur Identifikation wird das Verfahren der mehrfachen Integration benutzt, das sich insbesondere durch seine Robustheit gegenüber Meßrauschen auszeichnet [5]. Das identifizierte Systemmodell besitzt die Form

Modellansatz

$$G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0} e^{-T_t s}.$$

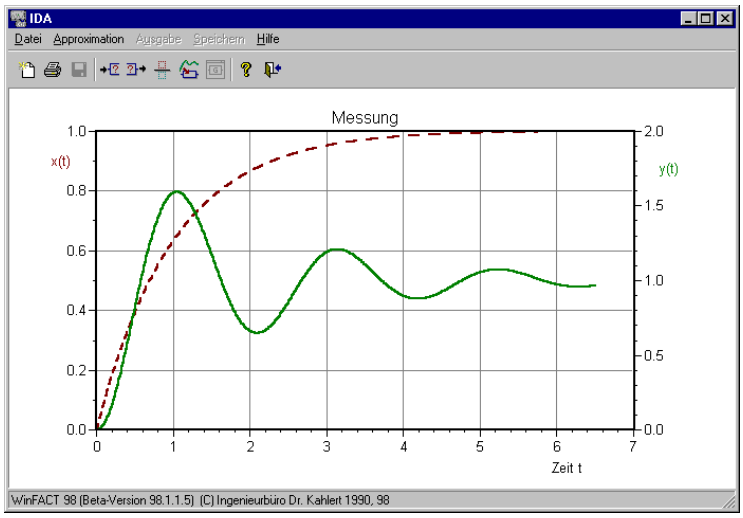
Zählergrad *m* und Nennergrad *n* können vom Anwender vorgegeben oder vom Programm automatisch ermittelt werden. Die Totzeit T_t wird ebenfalls - nach Eingabe einer Unter- und Obergrenze - automatisch vom Programm ermittelt.

Programmoptionen

Einlesen der Meßwerte

Die gemessenen Verläufe der Ein- und Ausgangsgröße müssen in getrennten Dateien vom Typ SIM vorliegen. Dabei ist für die Funktionsfähigkeit des Algorithmus wesentlich, daß beide Verläufe zu *denselben* Zeitpunkten aufgenommen wurden, beide Dateien also die gleiche Anzahl an Wertepaaren enthalten. Die Einhaltung dieser Bedingung wird vom Programm automatisch überprüft. Das Einlesen des Eingangsgrößenverlaufs erfolgt über die Menüfolge DATEI | EINGANGSSIGNAL X(T), die Tastenkombination <Strg><E> oder die Toolbar-Schaltfläche  , das Einlesen des Ausgangsgrößenverlaufs entsprechend über DATEI | AUSGANGSSIGNAL Y(T) bzw. <Strg><A> oder die Schaltfläche .

Die eingelesen Daten werden automatisch grafisch im Anwendungs-Hauptfenster dargestellt.

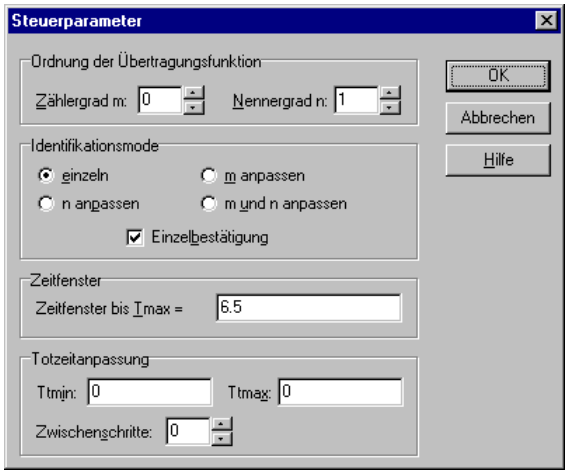


Hauptfenster nach dem Einlesen der Meßwerte: Die Eingangsgröße wird als gestrichelte rote Kurve, der Ausgangsgrößenverlauf als durchgezogene grüne Kurve dargestellt.

Die Qualität des ermittelten Modells ist ganz erheblich abhängig von der Anzahl der Meßwerte. Aus diesem Grunde sollten möglichst 500 Meßwerte oder mehr verwendet werden.

Steuerparameter für die Identifikation

Im Anschluß daran sind die Steuerparameter für die Identifikation über die Menüfolge DATEI | STEUERPARAMETER..., die Tastenkombination <Strg><S> oder die Toolbar-Schaltfläche festzulegen.



Eingabedialog für Steuerparameter

Der entsprechende Eingabedialog enthält folgende Daten:

Zählergrad m	(Maximaler) Zählergrad der Übertragungsfunktion
Nennergrad n	(Maximaler) Nennergrad der Übertragungsfunktion
Identifikationsmode	Wird der Identifikationsmode *Einzeln* gewählt, wird die Identifikation nur für die festgelegten Grade *m* und *n* durchgeführt. In den anderen Identifikationsmodi wird die optimale Übertragungsfunktion für alle Zähler- bzw. Nennergrade beginnend bei 0 bis zum angegebenen Wert ermittelt. Auf diese Art kann die geeignete Modellordnung automatisch bestimmt werden. Die automatische Identifikation kann vom Anwender jederzeit abgebrochen werden.

Zeitfenster bis Tmax

Das Zeitfenster ist standardmäßig so festgelegt, daß alle eingelesenen Daten zur Identifikation herangezogen werden. Soll speziell für kleinere Zeiten eine bessere Anpassung zwischen Modell und Messung erzielt werden, kann es angebracht sein, das Zeitfenster dementsprechend zu verkleinern. Dabei ist zu beachten, daß damit auch die Anzahl der zur Identifikation benutzten Meßwerte geringer wird, was in der Regel zu schlechteren numerischen Eigenschaften des Identifikationsalgorithmus führt.

Totzeitanpassung

Soll ein System mit Totzeit ermittelt werden, sind die Einstellungen im Gruppenfenster *Totzeitanpassung* vorzunehmen. Diese betreffen die verschiedenen Werte für die Totzeit T_t, für die eine Identifikation durchgeführt wird: $T_{t\,min}$ gibt den ersten Totzeitwert an, $T_{t\,max}$ den letzten Totzeitwert und *Zwischenschritte* die Anzahl der Zwischenwerte.

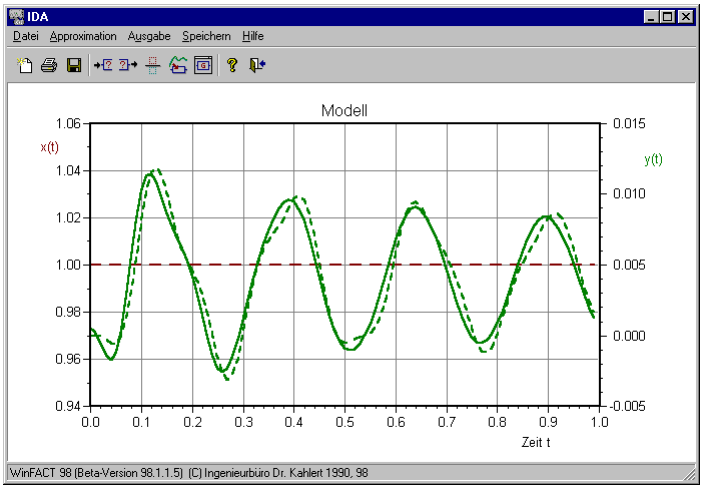
Beispiel: Für $T_{t\,min} = 0$, $T_{t\,max} = 1$ und vier Zwischenschritte werden für jede Zähler-/Nennergradkombination Identifikationen für Totzeiten von $T_t = 0,\ 0.2,\ 0.4,\ 0.6,\ 0.8$ und 1 durchgeführt. Die Totzeit, die zum besten Ergebnis führt, wird dann ausgewählt.

Identifikation

Nach Festlegung der Steuerparameter kann der Identifikationsvorgang über die Hauptmenüoption APPROXIMATION | APPROXIMATION oder die Toolbar-Schaltfläche  gestartet werden. Der Rechenzeitbedarf für einen Identifikationsvorgang hängt ab von der angesetzten Ordnung der Übertragungsfunktion, der Anzahl der eingelesenen Meßwerte und naturgemäß vom verwendeten Rechnertyp. In der Regel überschreitet er wenige Sekunden nicht. Bei automatischer Identifikation - womöglich noch mit unterlagerter Totzeitanpassung - erhöht sich der Rechenzeitaufwand dementsprechend.

Nach Beendigung wird das berechnete Modellverhalten automatisch angezeigt. Dargestellt werden

- der gemessene Eingangsgrößenverlauf (rote gestrichelte Kurve),
- der gemessene Ausgangsgrößenverlauf (grüne gestrichelte Kurve),
- der berechnete Ausgangsgrößenverlauf des identifizierten Modells (grüne durchgezogene Kurve).



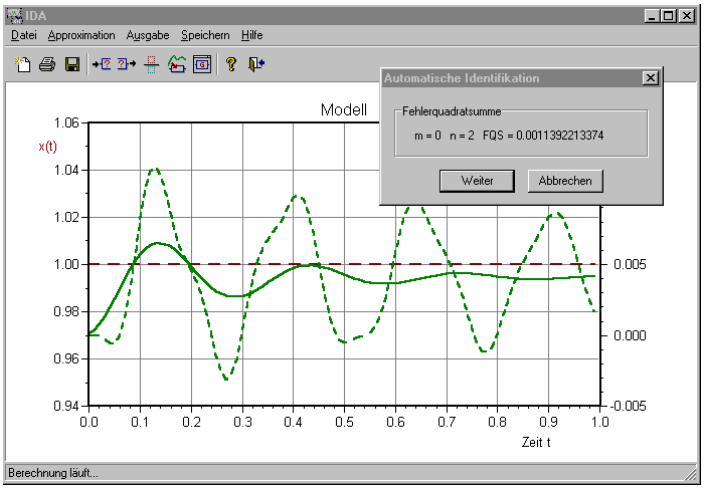
Grafische Anzeige der Ergebnisse

Bei Wahl eines anderen Identifikationsmodes als *Einzeln* wird nach jeder durchgeführten Identifikation das Ergebnis einerseits grafisch, andererseits in Form der ermittelten Fehlerquadratsumme ausgegeben. Diese ist definiert als

$$FQS = \sum_{i=1}^{m} \left(y_{i,Mess} - y_{i,Modell} \right)^2$$

Dabei ist:

m:	die Anzahl der eingelesenen Meßwerte
$y_{i,Mess}$:	der Meßwert zum Zeitpunkt t_i
$y_{i,Modell}$:	der berechnete Wert zum Zeitpunkt t_i



Automatische Identifikation

Die zugehörige Übertragungsfunktion kann über die Menüfolge AUSGABE | ÜBERTRAGUNGSFUNKTION ANZEIGEN oder die Schaltfläche auf dem Bildschirm ausgegeben werden. Außerdem kann sie, beispielsweise zur Weiterverarbeitung in anderen WinFACT-Modulen, durch AUSGABE | KOPIEREN bzw. <Strg><Einfg> in die Windows-Zwischenablage kopiert werden. Die entsprechende Fehlerquadratsumme kann mit Hilfe von AUSGABE | FEHLERQUADRATSUMME betrachtet werden.

Instabile Systeme

Speziell bei instabilen Systemen oder Systemen nahe der Stabilitätsgrenze kann das ermittelte Systemmodell je nach angesetztem Zähler- und Nennergrad instabil sein. Dies kann dazu führen, daß sich bei der nachfolgenden Simulation des Modellsystems sehr hohe Zahlenwerte für die Ausgangsgröße ergeben. In solchen Fällen wird die Simulation abgebrochen, um einen Zahlenüberlauf zu verhindern. Der Anwender wird durch eine entsprechende Warnmeldung darüber informiert. Obwohl in diesem Fall keine vollständige Simulation möglich ist, steht die ermittelte Übertragungsfunktion natürlich dennoch zur Verfügung.

Ein Abspeichern der Übertragungsfunktion ist über die Option SPEICHERN | SPEICHERN... oder die Schaltfläche möglich. Die zugehörige Datei ist vom Typ UFK.

Anwendung zur Modellreduktion

IDA läßt sich nicht nur zur Identifikation, sondern auch zur *Modellreduktion* heranziehen. Ziel der Modellreduktion ist es, ein System mit der Übertragungsfunktion

$$G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0}$$

zu ersetzen durch ein Modell niedrigerer Ordnung

$$\tilde{G}(s) = \frac{d_p s^p + d_{p-1} s^{p-1} + \ldots + d_1 s + d_0}{s^q + c_{q-1} s^{q-1} + \ldots + c_1 s + c_0}$$

mit

$$p + q < m + n \,,$$

das dem Originalmodell möglichst ähnlich ist:

$$G(s) \approx \tilde{G}(s) \,.$$

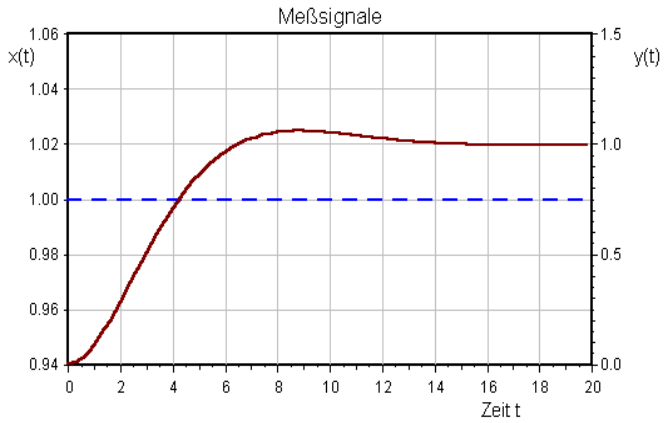
Zur Lösung dieser Aufgabe mit Hilfe von IDA berechnet man zunächst die Sprungantwort des Originalmodells (z. B. mit LISA) und legt die Zeitverläufe in entsprechenden Dateien vom Typ SIM ab. Diese Dateien werden dann in IDA eingelesen und das reduzierte Modell durch Vorgabe der gewünschten Grade *p* und *q* ermittelt.

Programmkonstanten

Maximale Anzahl an Meßwerten:	unbegrenzt
Maximale Modellordnung:	20

Anwendungsbeispiel

Es werde der folgende Verlauf von Eingangsgröße $x(t)$ und Ausgangsgröße $y(t)$ betrachtet (die zugehörigen Dateien befinden sich unter den Namen VZ2_X.SIM bzw. VZ2_Y.SIM im WinFACT-Beispiel-Verzeichnis):



Gemessene Zeitverläufe der Eingangsgröße x(t) (gestrichelt) und der Ausgangsgröße y(t) (durchgezogene Kurve)

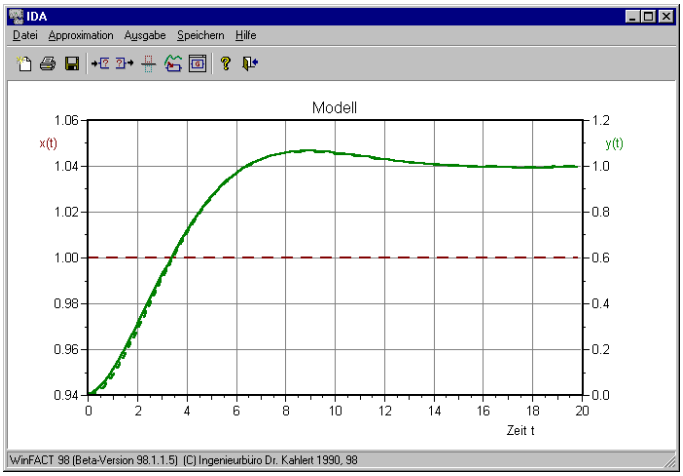
Das System weist typisches PT_2-Verhalten auf und soll daher durch einen Ansatz der Form

$$G(s) = \frac{b_0}{s^2 + a_1 s + a_0}$$

mit $m = 0$, $n = 2$ identifiziert werden. IDA ermittelt für die Übertragungsfunktion

$$G(s) = \frac{0.21}{s^2 + 0.59s + 0.21} .$$

Grafisch erkennt man unmittelbar die Güte des berechneten Modells.



Ergebnis der Identifikation des Verzögerungsgliedes 2. Ordnung

Im Beispiel-Verzeichnis befinden sich fünf weitere, komplexere Beispiele:

CHEN3_X.SIM, CHEN3_Y.SIM:

Beispielsystem mit $m = 6$ und $n = 8$, läßt sich aber bereits gut durch $m = 3$, $n = 4$ approximieren.

EX1_X.SIM, EX1_Y.SIM:

Beispielsystem aus [5] mit $m = 1$ und $n = 2$, die Eingangsgröße ist hier eine Exponentialfunktion.

N4_X.SIM, N4_Y.SIM:

Beispielsystem mit Allpaßanteil ($m = 2$ und $n = 4$).

RAUSCH_X.SIM, RAUSCH_Y.SIM:

Beispielsystem mit starkem Meßrauschen ($m = 0$ und $n = 2$).

SIN_X.SIM, SIN_Y.SIM:

Beispielsystem mit sinusförmiger Eingangsgröße ($m = 0$ und $n = 2$).

4 Analyse linearer Systeme mit LISA

Leistungsumfang

LISA ermöglicht die Analyse linearer Systeme mit der Eingangsgröße $u(t)$ und der Ausgangsgröße $y(t)$, die in Form einer gebrochen rationalen Übertragungsfunktion mit Totzeit

Systemmodell

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0} e^{-Ts}$$

vorliegen. Die Systemanalyse umfaßt folgende Punkte:

- Berechnung der Sprungantwort

  Es wird die Antwort des Systems auf eine sprungförmige Testfunktion

  $$u(t) = \begin{cases} 0 & \text{für } t < 0 \\ 1 & \text{für } t \geq 0 \end{cases}$$

  ermittelt. Simulationsdauer und -schrittzahl sind vorgebbar. Zur Simulation wird das Matrizenexponentialverfahren [3] benutzt.

- Darstellung des Frequenzgangs in Form des Bode-Diagramms

  Es wird der Frequenzgang

  $$G(\mathrm{j}\omega) = \frac{Y(\mathrm{j}\omega)}{U(\mathrm{j}\omega)} = \frac{b_m (\mathrm{j}\omega)^m + b_{m-1} (\mathrm{j}\omega)^{m-1} + \ldots + b_1 (\mathrm{j}\omega) + b_0}{(\mathrm{j}\omega)^n + a_{n-1} (\mathrm{j}\omega)^{n-1} + \ldots + a_1 (\mathrm{j}\omega) + a_0} e^{-\mathrm{j}\omega T}$$

  für einen vorgebbaren Frequenzbereich in Form der Betragskennlinie $|G(\mathrm{j}\omega)|_{\mathrm{dB}}$ und der Phasenkennlinie $\angle G(\mathrm{j}\omega)$ berechnet und über der logarithmischen Frequenzachse dargestellt.

- Darstellung des Frequenzgangs in Form der Nyquist-Ortskurve

  In diesem Fall wird der Frequenzgang aufgesplittet in Realteil $\mathrm{Re}\{G(\mathrm{j}\omega)\}$ und Imaginärteil $\mathrm{Im}\{G(\mathrm{j}\omega)\}$ und in der komplexen Ebene aufgetragen.

- Berechnung der Wurzelortskurve

Es wird der Verlauf der Eigenwerte des geschlossenen Regelkreises in Abhängigkeit von der Verstärkung eines P-Reglers ermittelt und in der komplexen Ebene dargestellt.

- Berechnung von Pol- und Nullstellen

  Es werden die Polstellen (Eigenwerte) und Nullstellen von $G(s)$ berechnet. Zur Anwendung kommt die Methode von Lagrange.

Das Programm verfügt über eine MDI-Schnittstelle, so daß ein- und dasselbe System gleichzeitig in verschiedenen Darstellungsformen, mit unterschiedlichen Skalierungen usw. dargestellt werden kann.

Programmoptionen

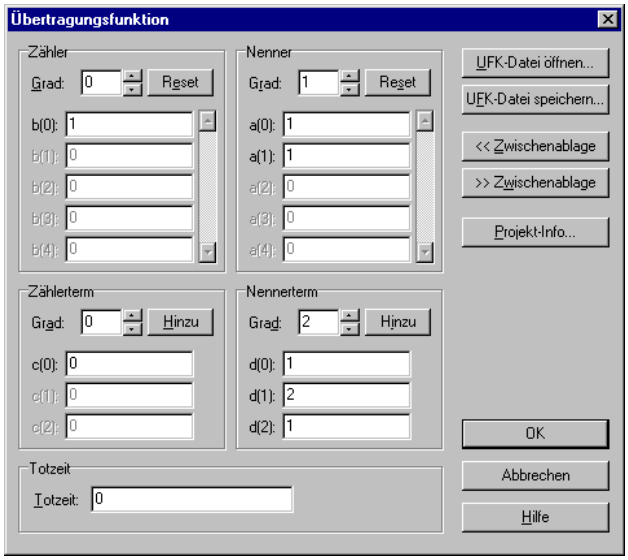
Einlesen der Daten

Nach dem Aufruf des Programms wird automatisch ein (zunächst noch leeres) Dokumentfenster geöffnet. Die zu analysierende Übertragungsfunktion kann wahlweise über die Tastatur, über die Zwischenablage oder aus einer Datei vom Typ UFK eingelesen und danach jederzeit modifiziert und abgespeichert werden. Zu diesem Zweck werden die Menüoptionen SYSTEMDATEI ÖFFNEN, SYSTEMDATEI SPEICHERN (UNTER) bzw. SYSTEMDATEI MODIFIZIEREN im Menü DATEI oder die entsprechenden Schaltflächen der Toolbar angewählt. Das entsprechende System wird dann im gerade aktiven Dokumentfenster dargestellt. Soll ein neues Dokumentfenster geöffnet werden, so ist zunächst die Menüoption DATEI | NEUES FENSTER anzuwählen. Das nachfolgende Bild zeigt den Eingabedialog für Übertragungsfunktionen. Dieser Eingabedialog ermöglicht

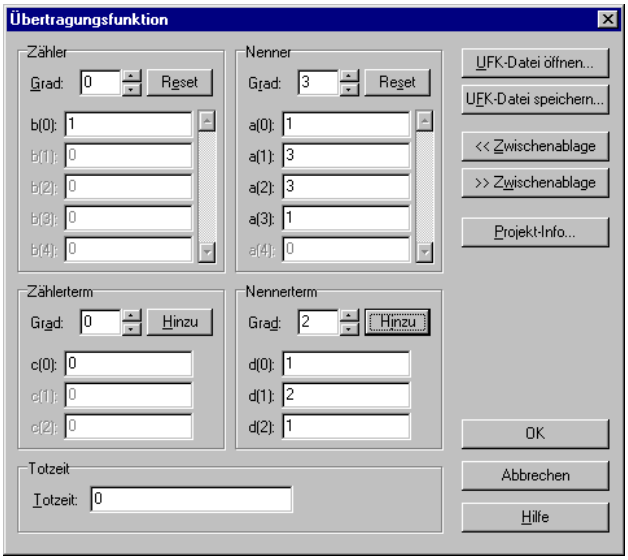
- die Modifikation der Übertragungsfunktion,
- das Kopieren der Übertragungsfunktion in die Zwischenablage,
- das Einfügen einer Übertragungsfunktion aus der Zwischenablage,
- das Abspeichern der Übertragungsfunktion in einer UFK-Datei,

- das Laden einer Übertragungsfunktion aus einer UFK-Datei,
- das Anzeigen und Modifizieren der Projekt-Information.

Zähler- und Nennerpolynom können - sofern sie bereits in ausmultiplizierter Form vorliegen - direkt eingegeben werden. Sie können jedoch alternativ auch in faktorisierter Form, d. h. als Kombination von Termen nullter bis zweiter Ordnung, aufgebaut werden. Diese Einzelterme werden in der unteren Hälfte des Dialogs eingegeben und über *Hinzu* jeweils dem aktuellen Gesamtzähler- bzw. -nennerpolynom hinzugerechnet. Über die *Reset*-Schaltflächen können Zähler- bzw. Nennerpolynom jederzeit auf eins zurückgesetzt werden. Die nachfolgenden Bildschirmfotos erläutern diese Vorgehensweise. Für die Berechnung der Wurzelortskurve wird eine eventuell vorhandene Systemtotzeit zu null gesetzt.



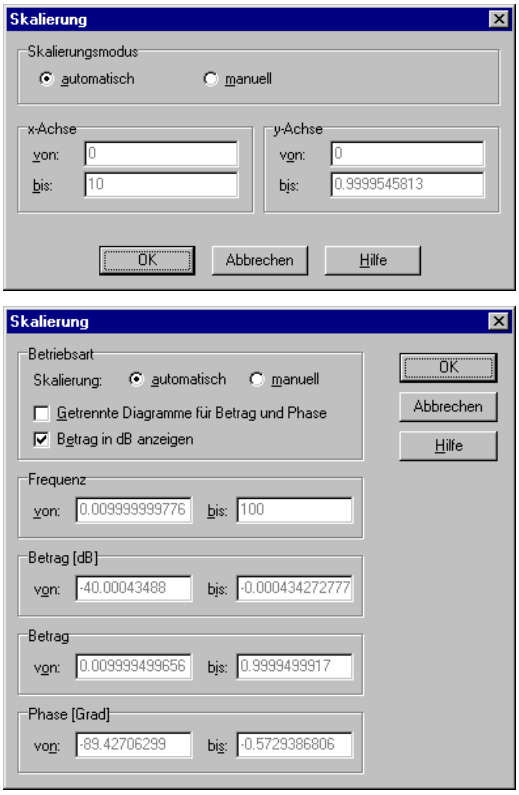
Beispiel für faktorisierte Vorgabe von Übertragungsfunktionen: Eingabe eines Nennerterms zweiter Ordnung...



... und Hinzufügen zum aktuellen Nennerpolynom

Darstellungsform und Speichern von Ergebnissen

Nach dem Aufruf eines neuen Dokumentfensters ist für dieses zunächst die Darstellungsform *Sprungantwort* eingestellt. Ein Wechsel der Darstellungsform für das aktive Dokumentfenster kann über das Menü ANZEIGE oder die entsprechenden Schaltflächen der Toolbar erfolgen. Sprungantworten, Bode-Diagramme und Ortskurven können in WinFACT-Dateien vom Typ SIM, BD bzw. OK abgespeichert werden. Dazu dient das Untermenü SPEICHERN. Ein Abspeichern von Wurzelortskurven und Pol-Nullstellen ist z. Z. noch nicht möglich. Die Skalierung von Koordinatenachsen sowie zusätzliche, von der Darstellungsform abhängige Parameter werden über die Menüoptionen SKALIERUNG bzw. PARAMETER beeinflußt.



Dialoge zur Skalierung linearer (oben) bzw. logarithmischer Diagramme (unten)

Die über den Parameter-Dialog einstellbaren Größen sind für die Sprungantwort:

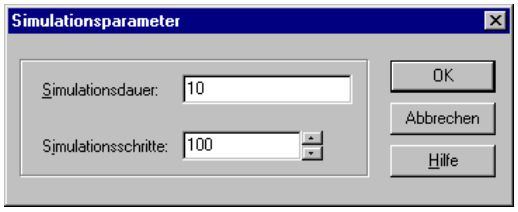
- Die Simulationsdauer T_{Ende}..
- Die Anzahl der Simulationsschritte n. Diese sollte so groß gewählt werden, daß die resultierende Schrittweite

$$\Delta T = \frac{T_{\text{Ende}}}{n-1}$$

maximal 10% der kleinsten Systemzeitkonstanten entspricht, um eine genügend genaue Simulation zu ermöglichen.

Voreingestellte Werte:

Simulationsdauer:	10
Simulationsschritte:	100



Dialog für Simulationsparameter

Für Bode-Diagramm und Ortskurve sind einstellbar:

- Die kleinste berechnete Frequenz ω_{min}
- Die Anzahl berechneter Frequenzdekaden
- Die Anzahl der Mindestwerte pro Dekade. Weist der Frequenzgang einen nicht hinreichend glatten Verlauf auf, kann dieser Wert u. U. vergrößert werden.
- Der Winkel $\Delta\Phi$, der angibt, wie stark sich der Phasenwinkel zwischen zwei Frequenzwerten maximal ändern darf, bevor zusätzliche Zwischenwerte berechnet werden. Weist der Frequenzgang einen nicht hinreichend glatten Verlauf auf, kann dieser Wert u. U. verkleinert werden.

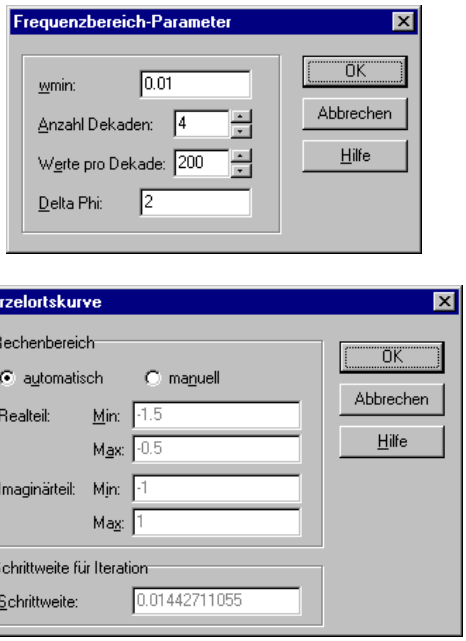
  Voreingestellte Werte:

ω_{min} *:*	0.01
Dekaden:	4
Werte/Dekade:	20
$\Delta\Phi$ *:*	5

Für die Wurzelortskurve sind wählbar:

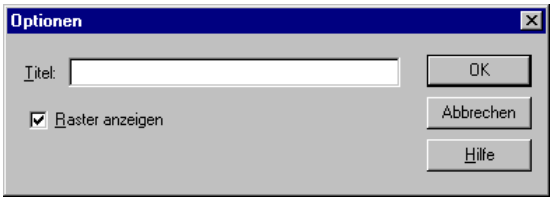
- Die *Berechnungsgrenzen* für die Wurzelortskurve (Real- und Imaginärteil). Bei der automatischen Berechnung werden diese abhängig von der Pol-Nullstellenverteilung des Systems festgelegt. Durch Anwahl von *manuell* können sie vom Anwender verändert werden.
- Die *Schrittweite* für die iterative Berechnung der Wurzelortskurve. Auch diese wird im automatischen Modus an die Pol-Nullstellenverteilung angepaßt. Bei Systemen mit sehr nah beieinanderliegenden

Polen und Nullstellen kann es sinnvoll sein, den von der Automatik ermittelten Wert manuell zu verkleinern; bei sehr weit auseinanderliegenden Werten kann er u. U. vergrößert werden.



Dialog für Frequenzbereichsparameter (oben) und Parameter der Wurzelortskurve (unten)

Der Menüpunkt OPTIONEN erlaubt schließlich für jedes Dokumentfenster die Eingabe eines Titel sowie das Zu- und Abschalten des Koordinatenrasters.



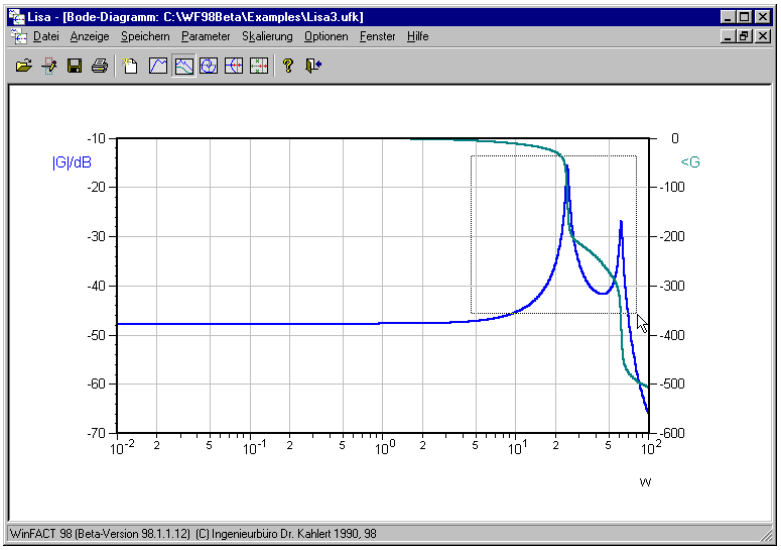
Optionen-Dialog

Zoom-Modus und Informationen in der Statuszeile

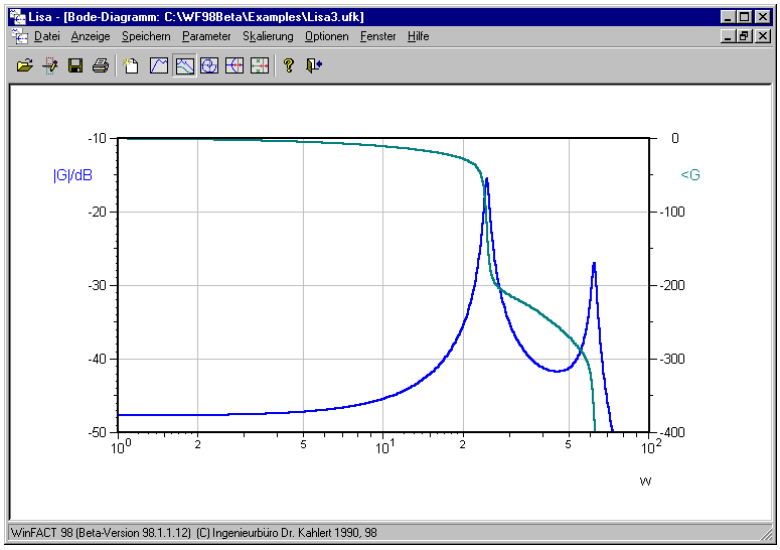
In allen Darstellungsformen mit Ausnahme der Pol-/Nullstellenausgabe ist ein *Zoom-Modus* verfügbar. Er ermöglicht eine vergrößerte Darstellung eines Bildausschnitts mit Hilfe der Maus. Dazu wird der linke obere Eckpunkt des zu vergrößernden Bildausschnitts mit der linken Maustaste angeklickt und dann bei festgehaltener Maustaste der Bereich festgelegt. Nach dem Loslassen der Maustaste erfolgt eine automatische Neuausgabe des vorgegebenen Bereiches. Dazu wird intern auf manuelle Skalierung umgeschaltet. Zur Wiederherstellung des kompletten Bereichs schaltet man daher entweder zurück auf automatische Skalierung oder betätigt den der Systemdarstellung entsprechenden Toolbar-Button.

In den Darstellungsarten *Nyquist-Ortskurve* und *Wurzelortskurve* werden in der Statuszeile zusätzliche Informationen angezeigt, sobald der Mauscursor sich auf der Kurve befindet:

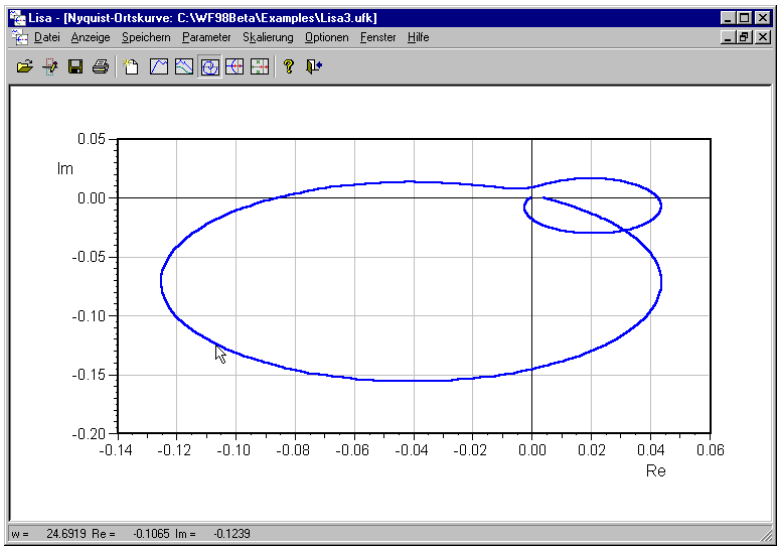
- Bei Nyquist-Ortskurven werden Frequenz, Real- und Imaginärteil des angewählten Kurvenpunktes angezeigt.
- Bei Wurzelortskurven werden Verstärkung, Real- und Imaginärteil angezeigt.



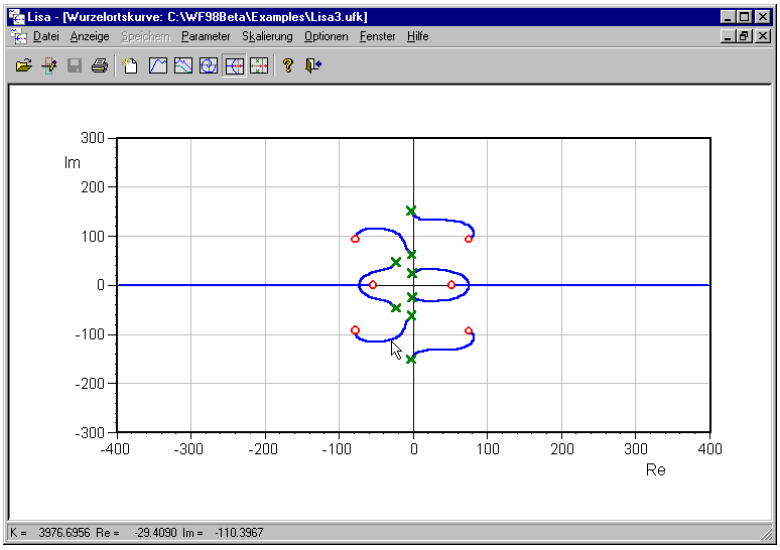
Zoom-Modus: Wahl des zu vergrößernden Bildausschnitts...



...und Anzeige des vergrößerten Bildausschnitts



Anzeige zusätzlicher Informationen in der Statuszeile: Nyquist-Ortskurve



Anzeige zusätzlicher Informationen in der Statuszeile: Wurzelortskurve

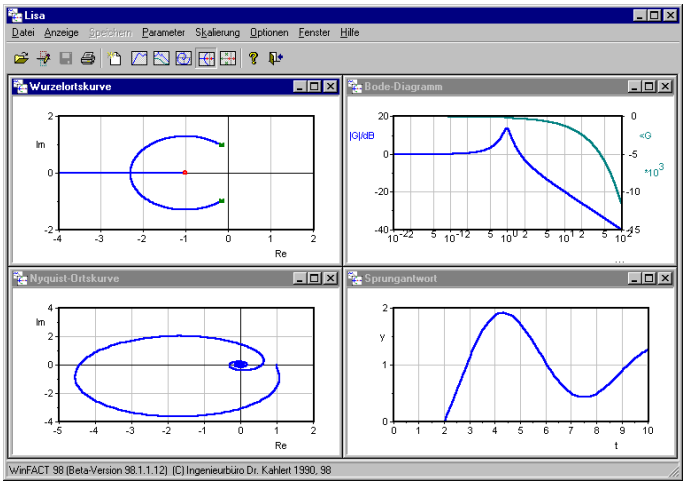
Programmkonstanten

Maximale Systemordnung: 20

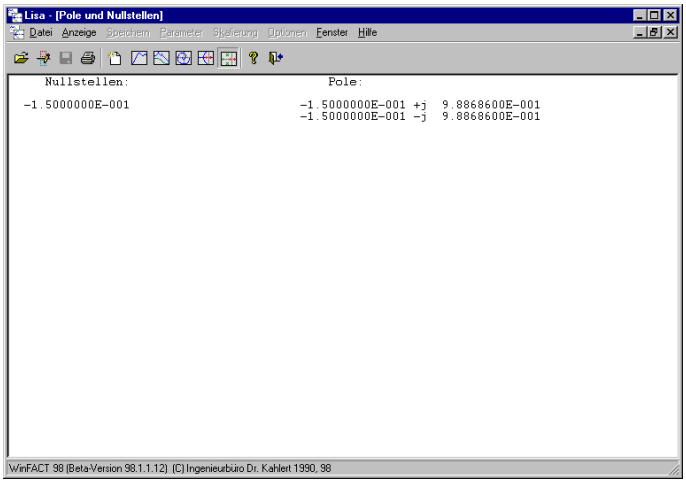
Anwendungsbeispiel

Die folgenden Bildschirmgrafiken zeigen die Darstellung von Sprungantwort, Bode-Diagramm, Ortskurve und Wurzelortskurve sowie Pol-/Nullstellen für ein System 2. Ordnung mit der Übertragungsfunktion

$$G(s) = \frac{s+1}{s^2+0.3s+1} e^{-2s}.$$



Sprungantwort, Bode-Diagramm, Nyquist-Ortskurve und Wurzelortskurve für Beispiel



Ausgabe von Polen und Nullstellen für Beispielsystem

Im Beispiel-Verzeichnis befinden sich zusätzlich folgende Beispieldateien:

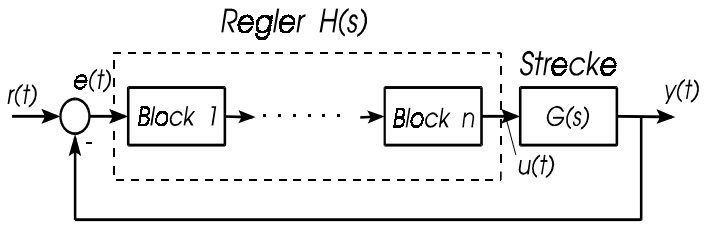
LISA1.UFK:	System mit $m = 1$, $n = 2$ und Allpaßverhalten
LISA2.UFK:	Entspricht LISA1.UFK mit zusätzlicher Totzeit
LISA3.UFK:	System mit $m = 6$, $n = 8$ und Allpaßverhalten

5 Entwurf linearer Regelkreise mit RESY

Leistungsumfang

RESY ermöglicht die Analyse, Synthese und Simulation linearer einschleifiger Regelkreise der folgenden Struktur:

Regelkreis-struktur



Zugrundeliegende Regelkreisstruktur

Die Regelstrecke muß in Form einer Übertragungsfunktion $G(s)$ vorliegen. Der Regler kann schrittweise aus linearen Standardkomponenten aufgebaut werden. Zur Verfügung stehen:

- P-, I-, PI-, PD- und PID-Komponenten der allgemeinen Form

$$H_i(s) = K_R\left(1 + \frac{1}{T_N s} + \frac{T_V s}{1 + T_{Vz} s}\right)$$

- Lead-Lag-Glieder der Form

$$H_{i,\mathrm{Lead}}(s) = \frac{1 + \dfrac{s}{\omega_i}}{1 + \dfrac{s}{m\omega_i}} \quad \textit{bzw.} \quad H_{i,\mathrm{Lag}}(s) = \frac{1 + \dfrac{s}{m\omega_i}}{1 + \dfrac{s}{\omega_i}}$$

- Allgemeine gebrochen rationale Übertragungsfunktionen mit Totzeit der Form

$$H_i(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0} e^{-Ts}$$

RESY ermittelt daraus

- die Gesamtübertragungsfunktion $H(s)$ des Reglers,
- die Übertragungsfunktion $L(s) = G(s)H(s)$ des offenen Regelkreises,
- die Übertragungsfunktion $T(s) = L(s) / (1 + L(s))$ des geschlossenen Regelkreises,
- die zugehörigen Frequenzgänge $G(j\omega)$, $H(j\omega)$, $L(j\omega)$, $T(j\omega)$

und im Zeitbereich für eine sprungförmige Führungsgröße $r(t)$ den Verlauf

- der Regelabweichung $e(t)$,
- der Stellgröße $u(t)$,
- der Regelgröße $y_T(t)$

sowie die Sprungantwort $y_G(t)$ der Regelstrecke selbst.

Sowohl im Zeitbereich als auch im Frequenzbereich können charakteristische Kenngrößen ermittelt werden, die einen Anhaltspunkt für das dynamische Verhalten des Systems darstellen. Dies sind im Zeitbereich:

Kennwerte

- Die Überschwingweite M_p der Regelgröße. Sie entspricht dem erreichten Maximalwert der Regelgröße während des Ausregelvorgangs.
- Die Ausregelzeit T_a bezogen auf einen 10%-Fehlerschlauch um den stationären Endwert der Regelgröße. Sie ist ein Maß für die Schnelligkeit des Ausregelvorgangs.
- Die bleibende Regelabweichung $e(t \to \infty)$.
- Der maximale Stellgrößenbedarf u_{max}. Er entspricht dem Maximalwert des Betrags der Stellgröße $u(t)$ während des Ausregelvorgangs.

Im Frequenzbereich für den offenen Kreis:

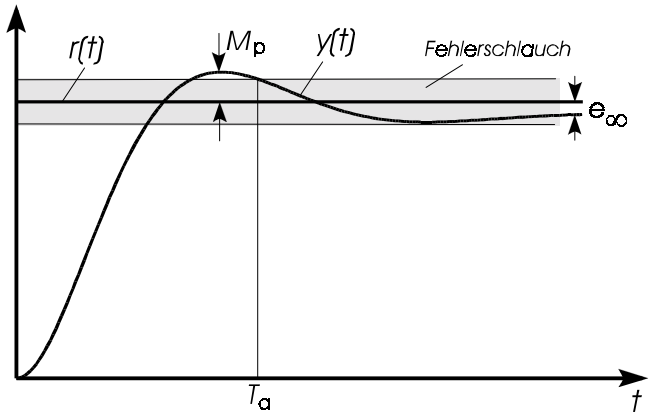
- Die Durchtrittsfrequenz ω_c, d. h. diejenige Frequenz, an der die Betragskennlinie die 0 dB-Linie schneidet.

- Die Phasenreserve Φ_r, die den Abstand der Phasenkennlinie von der -180°-Linie bei der Durchtrittsfrequenz darstellt. Sie ist ein Maß für das Schwing- bzw. Stabilitätsverhalten des geschlossenen Regelkreises.
- Der Amplitudenrand (Stabilitätsgrenze) A_r, der den negativen Betrag des offenen Kreises an der Stelle angibt, an der die Phasenkennlinie die -180°-Linie schneidet.

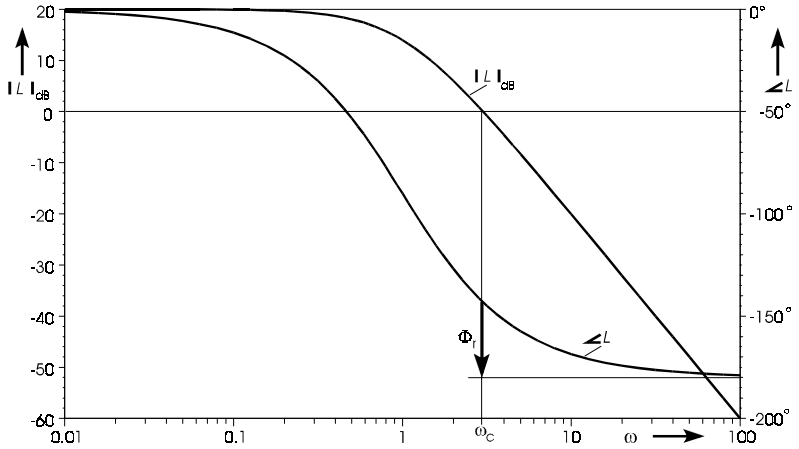
Für den geschlossenen Kreis:

- Die Bandbreite ω_b, die die Frequenz angibt, bei der die Betragskennlinie des geschlossenen Kreises die -3 dB-Linie schneidet.
- Die Resonanzüberhöhung M_m, d. h. der Maximalwert der Betragskennlinie des geschlossenen Kreises.

Der Frequenzgang kann wahlweise in Form des Bode-Diagramms oder der Nyquist-Ortskurve dargestellt werden. Die nachfolgenden Bilder verdeutlichen die Bedeutung einiger der angesprochenen Gütemaße (siehe auch [6, 7]).



Gütekriterien im Zeitbereich



Kennwerte im Frequenzgang des offenen Kreises

Programmoptionen

Dateiformat

RESY-Datensätze werden in Dateien vom Typ UFK abgespeichert. Sie unterscheiden sich vom Standard-UFK-Dateityp dadurch, daß nach der Übertragungsfunktion $G(s)$ der Regelstrecke, die sich am Anfang der Datei befindet, zusätzlich alle Reglerblöcke abgespeichert werden. Jeder Reglerblock besteht aus seinem Bezeichner (z. B. LEAD) und den zugehörigen Blockparametern (z. B. m und ω_i).

```
0
2
 7.120000000000000E-0001
 7.120000000000000E-0001
 4.270000000000000E+0000
 1.000000000000000E+0000
0.00
PI
 5.000000000000000E+0000
1.100000000000000E+0000
```

Beispiel für eine RESY-Eingabedatei mit Regelstrecke und PI-Regler

Bildschirmaufbau

Nachfolgendes Bild zeigt ein typisches Hauptfenster während der Arbeit mit RESY. Der Darstellungsmodus des Regelkreisverhaltens kann über die Menüoption *Anzeige* mit der entsprechenden Unteroption bzw. des entsprechenden Tastenkürzels oder die Toolbar gewählt werden. Mögliche Optionen sind

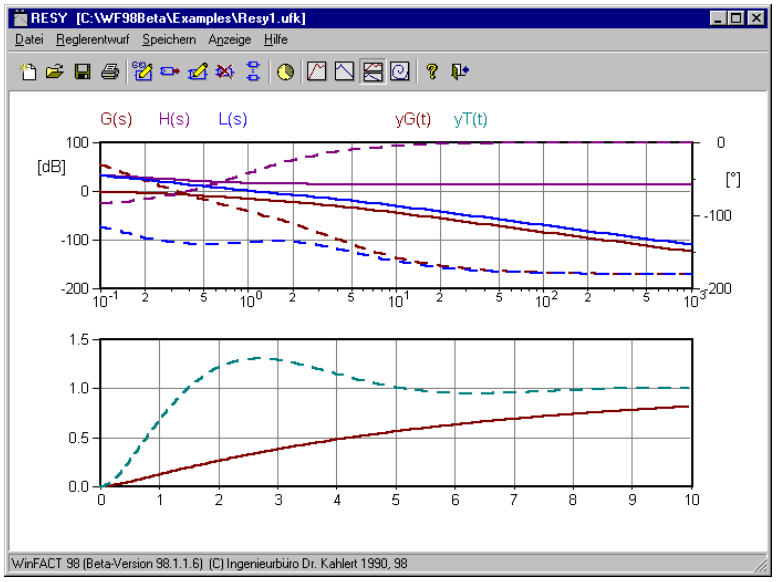
- Vollbild-Anzeige im Zeitbereich,
- Vollbild-Anzeige im Frequenzbereich als Bode-Diagramm oder Nyquist-Ortkurve,
- gleichzeitige Anzeige von Zeitverhalten und Bode-Diagramm (voreingestellt).

Die Auswahl der darzustellenden Frequenzgänge bzw. Zeitverläufe erfolgt über die Menüfolge ANZEIGE | OPTIONEN bzw. die Toolbar. Standardmäßig werden angezeigt

- im Frequenzbereich $G(s)$, $H(s)$ und $L(s)$,
- im Zeitbereich die Sprungantwort $y_G(t)$ der Regelstrecke sowie die Sprungantwort $y_T(t)$ des geschlossenen Regelkreises.

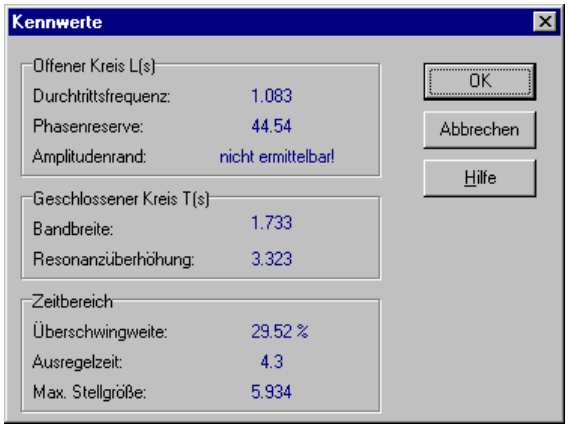
Im Anzeigemodus *Nyquist-Ortskurve* wird lediglich $L(s)$ angezeigt.

Die Skalierung der Koordinatenachsen kann über die Menüfolge DATEI | FREQUENZBEREICH bzw. DATEI | SIMULATIONSPARAMETER beeinflußt werden.



Hauptfenster des Programms (hier mit Anzeige von Zeit- und Frequenzbereich)

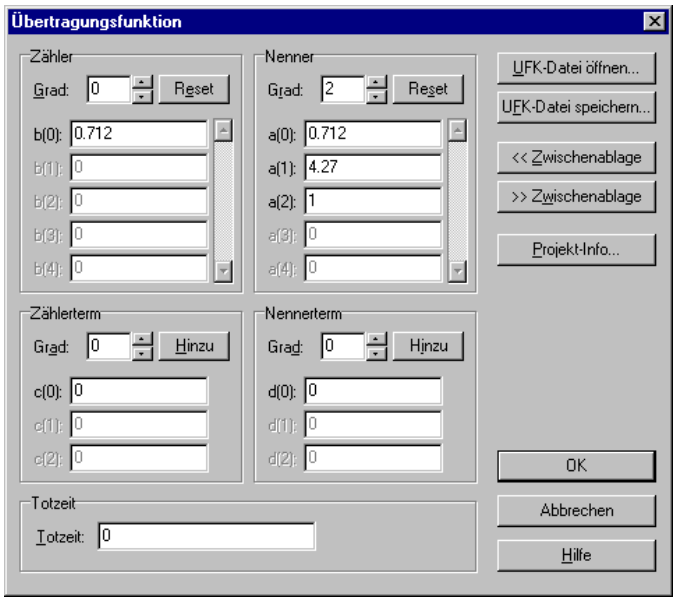
Die aktuellen Kennwerte im Zeit- und Frequenzbereich werden über die Menüfolge REGLERENTWURF | KENNWERTE abgerufen. Sie werden dann in einem entsprechenden Dialogfenster angezeigt. Ist ein Kennwert nicht ermittelbar, so wird ein entsprechender Hinweis ausgegeben.



Ausgabe der Kennwerte

Konfigurierung des Regelkreises

Zur Konfigurierung des Regelkeises ist zunächst die Regelstrecke zu definieren bzw. aus einer Datei vom Typ UFK zu lesen. Die manuelle Eingabe bzw. Modifikation ist über die Menüfolge DATEI | REGELSTRECKE BEARBEITEN bzw. die Toolbar möglich. Man gelangt auf diese Weise in den nachfolgend dargestellten Eingabedialog. Über diesen Eingabedialog kann die Regelstrecke auch aus der Zwischenablage geholt werden (oberste Schaltfläche) bzw. in ihr abgelegt werden. Einzelheiten zum Eingabedialog entnehme man der Beschreibung zum Modul LISA.



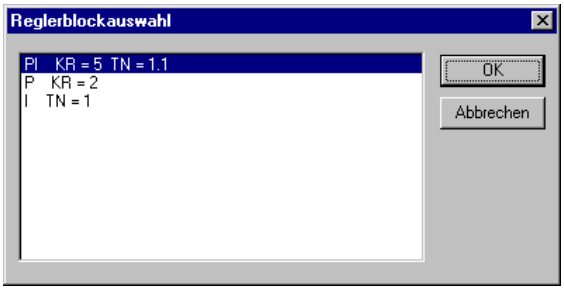
Eingabedialog für Regelstrecke

Nach Festlegung der Regelstrecke kann der Regler schrittweise aufgebaut werden. Dazu stehen im Menü REGLERENTWURF bzw. in der Toolbar folgende Optionen zur Verfügung:

- Einfügen von Reglerblöcken,
- Löschen von Reglerblöcken,
- Modifizieren bereits vorhandener Blöcke,

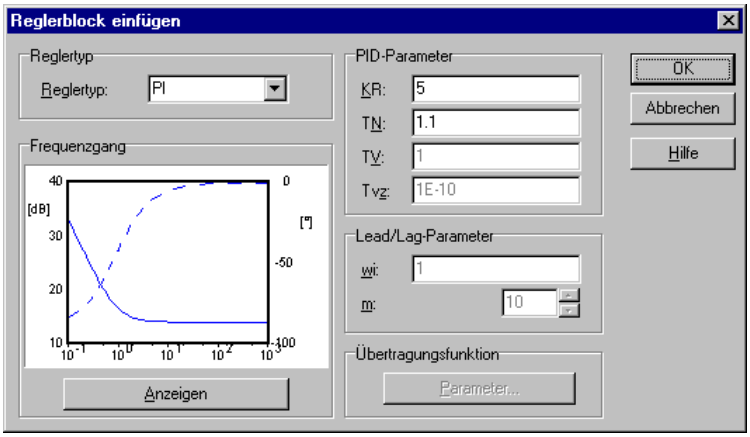
- Auflisten aller vorhandenen Blöcke.

Die Auswahl des jeweiligen Reglerblocks bei den Optionen *Löschen* und *Modifizieren* wird über eine Liste vorgenommen.



Anwahl bereits vorhandener Reglerblöcke über eine Listbox

Nachfolgendes Bild zeigt den zentralen Eingabedialog für die Konfigurierung von Reglerblöcken.



Zentraler Eingabedialog für die Konfigurierung von Reglerblöcken

Der Dialog enthält:

- Ein Gruppenfeld zur Festlegung des *Reglertyps* (linker Rand des Dialogfensters)
- Editierfelder zur Eingabe der *Reglerparameter* (Mitte des Dialogfensters)

- Ein Anzeigefenster für den *Frequenzgang* des konfigurierten Reglerblocks

Die Schaltfläche *Parametereingabe* dient zur Festlegung von beliebigen Regler-Übertragungsfunktionen. Sie ist bei anderen Reglertypen nicht anwählbar. Bevor der entsprechende Reglerblock in den Regelkreis übernommen wird, kann der Frequenzgang des Blocks über die Schaltfläche *Anzeigen* berechnet und angezeigt werden. Die Betragskennlinie wird in diesem Fall als durchgezogene Kurve, die Phasenkennlinie gestrichelt dargestellt.

Speichern von Ergebnissen

Über das Menü SPEICHERN ist ein Abspeichern sämtlicher Ergebnisse in entsprechenden Dateien möglich. Es können gespeichert werden

- die Übertragungsfunktionen $H(s)$, $L(s)$ und $T(s)$ in Dateien vom Typ UFK,
- die Zeitverläufe $y_G(t)$ und $y_T(t)$ in Dateien vom Typ SIM,
- die Frequenzgänge $H(j\omega), G(j\omega), L(j\omega)$ und $T(j\omega)$ in Dateien vom Typ BD oder OK, abhängig vom aktuellen Anzeigemodus.

Programmkonstanten

Maximale Ordnung des offenen Kreises:	20
Maximale Anzahl an Reglerblöcken:	20
Maximale Anzahl an Simulationsschritten:	unbegrenzt

Anwendungsbeispiel

Für die Regelstrecke mit der Übertragungsfunktion

$$G(s) = \frac{0.712}{s^2 + 4.27s + 0.712}$$

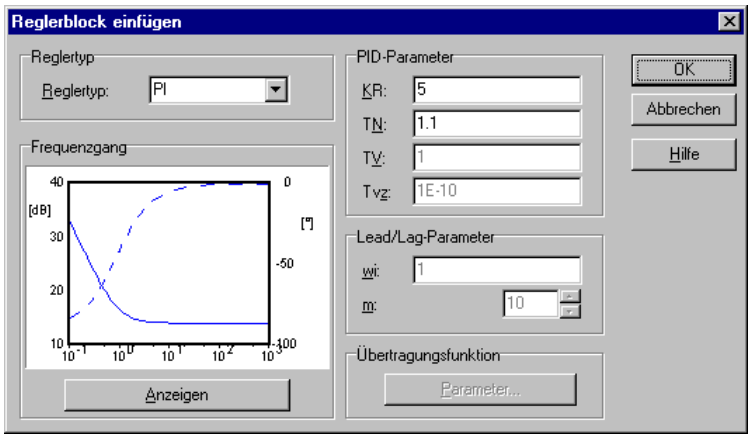
ist ein PI-Regler zu entwerfen, der für den Frequenzgang $L(\mathrm{j}\omega)$ die folgenden Spezifikationen erfüllt:

- Die Durchtrittsfrequenz soll $\omega_c \approx 1$ betragen.
- Die Phasenreserve soll $\Phi_r \approx 45°$ betragen.

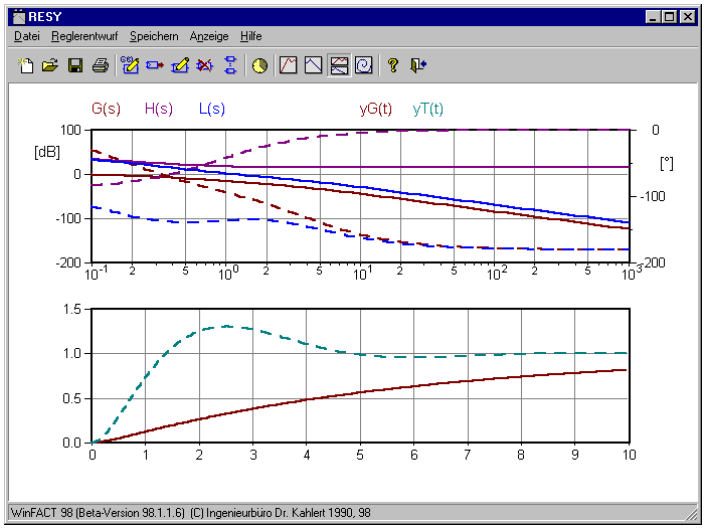
Der Entwurf liefert für den resultierenden Regler die Parameter

$$K_R = 5, \quad T_N = 1.1$$

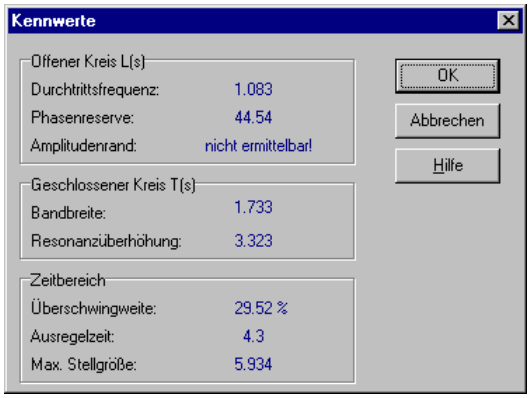
Die nachfolgenden Bilder zeigen die Konfigurierung des Reglers sowie das Hauptfenster des Programms nach Konfigurierung des Reglers. Die Kennwertermittlung liefert eine Überschwingweite von ungefähr 30% und eine Ausregelzeit von 4.3 bei einer maximalen Stellgröße von 5.9. Das Beispiel befindet sich unter dem Namen RESY1.UFK im Beispiel-Verzeichnis.



Konfigurierung des PI-Reglers



Entwurf eines PI-Reglers für Strecke 2. Ordnung



Zugehörige Kennwerte

6 Simulation und Synthese von Zustandsraumsystemen mit SUSY

Leistungsumfang

Das WinFACT - Modul SUSY (**S**imulation **u**nd **Sy**nthese im Zustandsraum) ermöglicht die Behandlung von Eingrößensystemen in Zustandsraumdarstellung

$$\dot{\underline{x}} = \underline{A}\underline{x} + \underline{b}u,\ y = \underline{c}^{\mathrm{T}}\underline{x} + du\ ,$$

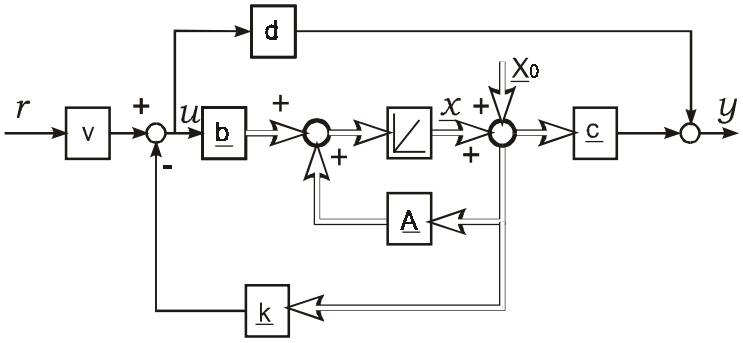
die gegebenenfalls über einen linearen Zustandsregler der Form

$$u = -\underline{k}^{\mathrm{T}}\underline{x} + Vr$$

zu einem geschlossenen Regelkreis ergänzt werden können. Dabei sind:

$\underline{A}$	die Systemmatrix des offenen Kreises
$\underline{b}$	der Eingangsvektor
$\underline{c}$	der Ausgangsvektor
d	der Durchgriff
$\underline{x}_0$	die Zustandsgrößen zum Zeitpunkt $t = 0$
V	der Vorfilter, der dafür sorgt, daß die Ausgangsgröße im stationären Zustand mit der Führungsgröße übereinstimmt
$\underline{k}$	der Rückführungsvektor (Regler)

Der betrachtete Zustandsregelkreis weist demnach folgende Struktur auf:



Geschlossener Zustandsregelkreis

SUSY erleichtert Ihnen die Arbeit durch folgende Merkmale:

- Einfache und übersichtliche Eingabe von Systemen
- Parallele Bearbeitung mehrerer Systeme
- Darstellung der Simulationsergebnisse durch Trajektorienfeld, Einzeltrajektorien oder Zeitverlauf
- Schnelle Umschaltung zwischen geschlossenem und offenem Regelkreis
- Eigenwertermittlung von offenem und geschlossenem Regelkreis
- Synthese von Reglern durch Riccati-Entwurf, Polplazierung oder manuelle Reglervorgabe
- Dokumenterzeugung über die Reglersynthese im ASCII-Format

Nachfolgend werden alle Möglichkeiten von SUSY am Beispiel

$$\underline{\dot{x}}(t) = \begin{pmatrix} 0 & -6 \\ -1 & -5 \end{pmatrix} \cdot \underline{x}(t) + \begin{pmatrix} 0 \\ 2 \end{pmatrix} \cdot u(t), \qquad y(t) = \begin{pmatrix} 2 & 0 \end{pmatrix} \cdot \underline{x}(t) + (0.5) \cdot u(t)$$

erläutert.

Systemein-/ausgabe

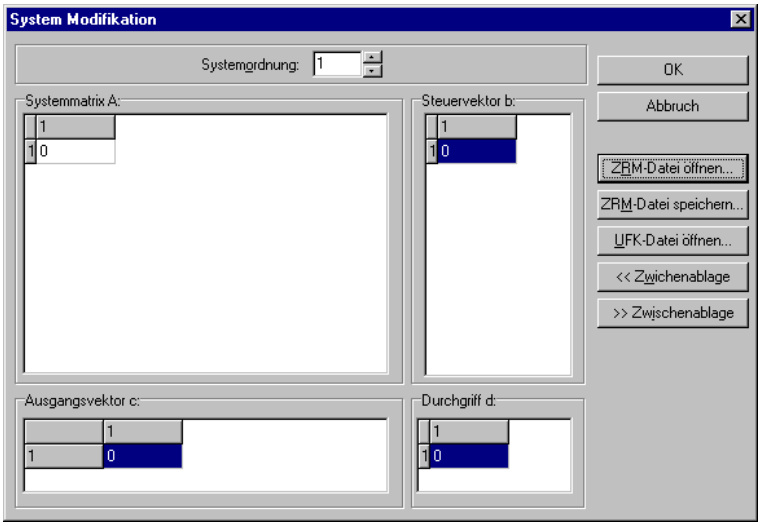
In diesem Abschnitt werden Sie mit den verschiedenen Möglichkeiten zur Systemeingabe vertraut gemacht. Dabei wird auf die manuelle Eingabe und die Eingabe durch Transformation einer Übertragungsfunktion eingegangen. Im Abschnitt *Systemausgabe* werden die Möglichkeiten der einzelnen Ausgabeformen besprochen.

Systemeingabe

Manuelle Systemeingabe

Bevor Sie in die manuelle Eingabe gelangen, müssen Sie ein neues Anzeigefenster in SUSY öffnen. Betätigen Sie dazu den Menüpunkt DATEI | NEU oder die entsprechende Schaltfläche der Werkzeugleiste. Anschließend kann über DATEI | SYSTEM MODIFIZIEREN das System im folgenden Dialog eingegeben werden. Um unser Beispielsystem einzugeben, müssen Sie zunächst die Systemordnung auf zwei erhöhen. Anschließend sind die benötigten Felder anwählbar. Nach der Eingabe sollten Sie Ihr Zustandsraummodell durch Betätigen der Schaltfläche *ZRM-Datei speichern* speichern. Das Laden und Speichern innerhalb des Dialoges bezieht sich nur auf das Zustandsraumsystem selbst. Ein Laden bzw. Speichern aus dem Menü (DATEI | SYSTEMDATEI ÖFFNEN bzw. DATEI | SYSTEMDATEI SPEICHERN und DATEI | SYSTEMDATEI SPEICHERN UNTER) oder der Werkzeugleiste speichert dagegen auch

- die zuletzt verwendete Reglerentwurfsmethode,
- die Parameter der Entwurfsmethoden und
- ob das geschlossene oder offene System zuletzt angezeigt wurde.



System-Modifikations-Dialog

Die Statuszeile zeigt Ihnen stets die Ordnung Ihres Systems an und gibt Auskunft, ob es sich momentan um das geschlossene oder um das offene System handelt.

Einlesen von Übertragungsfunktionen

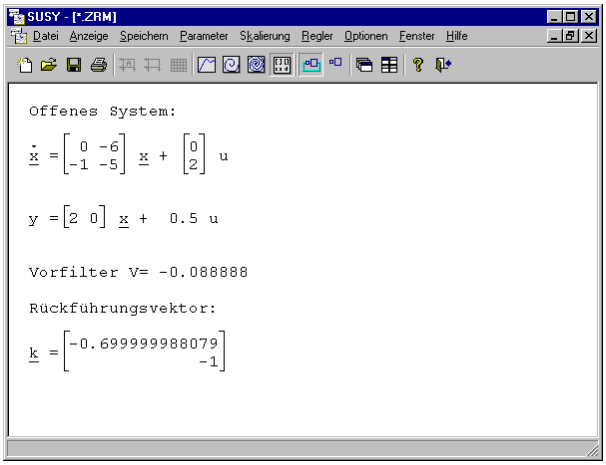
Über die Schaltfläche *UFK-Datei öffnen* im Dialog zur Systemmodifikation können Sie Übertragungsfunktionen in Zustandsraummodelle transformieren. Sie erhalten dabei die Regelungsnormalform oder auch Frobenius-Form des Systems. Für die nähere Beschreibung dieser Form sei auf [7] verwiesen.

Systemausgabe

Die Systemausgabe erfolgt in Matrizenschreibweise. Dabei können Sie im Dialog *Systemanzeige* (OPTIONEN | SYSTEMANZEIGE...) einstellen, was bei Anwahl von ANZEIGE | SYSTEMANZEIGE zu sehen sein soll.

Dialog zur Parametrierung der Systemanzeige

Teile des geschlossenen Systems und der Rückführungsvektor können nur angezeigt werden, wenn ein Regler für das System entworfen wurde.



Systemanzeige mit Rückführungsvektor und Vorfilter

Simulation

In diesem Kapitel werden die verschiedenen Möglichkeiten der Simulation besprochen. Dabei erfahren Sie, welche Parameter, Optionen und sonstigen Einstellungen gewählt werden können.

Simulationsparameter

Um die Simulationsparameter ändern zu können, müssen Sie zunächst zu einer der folgenden Anzeigen wechseln:

- Zeitverlauf
- Trajektorie
- Trajektorienfeld

Anschließend ist das Menü PARAMETER aktiv. Wählen Sie den Punkt SIMULATION aus, so erhalten Sie den Dialog zum Ändern der Simulationsparameter.

Dialog zum Ändern der Simulationsparameter

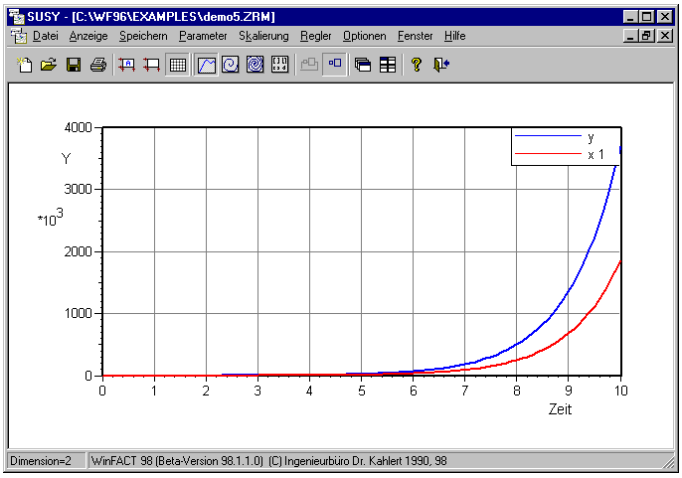
In diesem lassen sich verschiedene Eingangsfunktionen, die Endzeit der Simulation und die Anzahl der Schritte (und somit die Schrittweite) einstellen (zur Definition einer Funktion über den Funktionsparser siehe Kapitel 10, Beschreibung des Generator-Blocks). Des weiteren haben Sie die Möglichkeit, sich den maximalen Stellgrößenbedarf bei geschlossenen Systemen ausgeben zu lassen. Beachten Sie dabei, daß dieser nur innerhalb der Simulationszeit bestimmt werden kann.

Zeitverlauf

Die Menüfolge ANZEIGE | ZEITVERLAUF oder das Betätigen der entsprechenden Schaltfläche in der Werkzeugleiste simuliert einen Zeitverlauf des gewählten Zustandsregelkreises. Je nach Ordnung des Systems haben Sie die Möglichkeit, entsprechend viele Zeitverläufe der Zustandsgrößen anzeigen zu lassen. Dazu verwenden Sie den Menüpunkt PARAMETER | ANZEIGE.

Da wir in unserem Beispiel zwei Zustandsgrößen haben, sind in der Liste der möglichen Zeitverläufe nur zwei Größen eingetragen, von denen nur *y* angezeigt wird. Bei Systemen höherer Ordnung würde die Liste mit *x3* usw. fortgesetzt werden. Klicken Sie nun in der Liste der anzuzeigenden Zeitverläufe auf das Kästchen vor *x1* und verlassen den Dialog mit *OK*, so erhalten Sie in Ihrem Fenster die gewünschten beiden Zeitverläufe.

Dialog zur Parametrierung des Zeitverlaufs

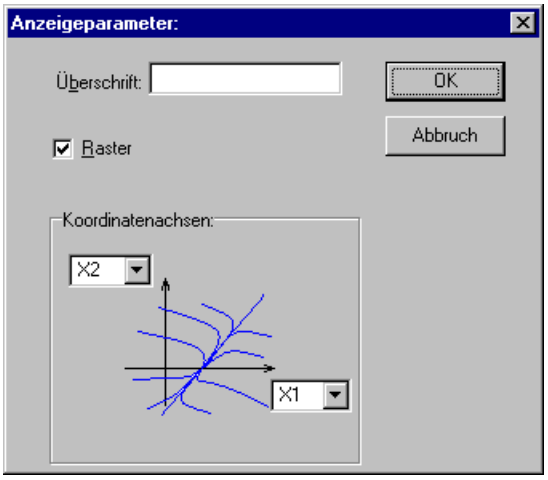


Zeitverläufe x_1 und y des offenen Beispielregelkreises

Trajektorien

Einzeltrajektorien

Der Menüpunkt ANZEIGE | EINZELTRAJEKTORIE ermittelt eine Trajektorie des Systems. Durch PARAMETER | ANZEIGE können Sie in dem entsprechenden Dialog auswählen, welche zwei Zustandsgrößen bei der Simulation als Trajektorie dargestellt werden.



Dialog zu Parametrierung von Trajektorienanzeigen

Trajektorien werden in drei verschiedenen Farben dargestellt:

grün	Trajektorie, die bei den Systemkoordinaten zum Zeitpunkt $t = 0$ startet
rot	selektierte Trajektorie (siehe *Trajektorien auswählen*)
blau	sonstige Trajektorie

Trajektorienfelder

Trajektorienfelder erzeugen Sie durch die Menüoption ANZEIGE | TRAJEKTORIENFELD, wobei Sie zwei weitere Möglichkeiten haben:

1. DURCH TRAJEKTORIE IN T(0)*:* Es wird eine Trajektorie simuliert, die die Koordinaten des System zum Zeitpunkt $t = 0$ als Anfangspunkt hat. Diese Trajektorie bestimmt das neue Koordinatensystem. Anschließend

werden die Startpunkte der Trajektorien so bestimmt, daß sie, im vernünftigen Abstand zueinander, auf den Rändern des jetzigen Koordinatensystems liegen. Diese Einstellung ist auch durch eine Schaltfläche in der Werkzeugleiste erreichbar.

2. AUS MOMENTANEN KOORDINATEN: Es wird ein Trajektorienfeld in die vorhandene Koordinatenebene gezeichnet. Die Startwerte der Trajektorien liegen dabei auf den Rändern des Koordinatensystems.

Trajektorien auswählen

Es kann wünschenswert sein, einige Trajektorien zu selektieren, wenn beispielsweise das Trajektorienfeld zu dicht ist und einige aus diesem gelöscht werden sollen. Klicken Sie dazu einfach mit der linken Maustaste die entsprechenden Trajektorien an. Die durch diese Koordinate (genauer gesagt, wird ein Rechteck von 5×5 Punkten genommen) verlaufenden Trajektorien werden anschließend rot dargestellt. Sie sind somit selektiert.

Trajektorien löschen

Sie können ausgewählte Trajektorien löschen, indem Sie den Menüpunkt OPTIONEN | TRAJEKTORIEN LÖSCHEN verwenden.

Trajektorien hinzufügen

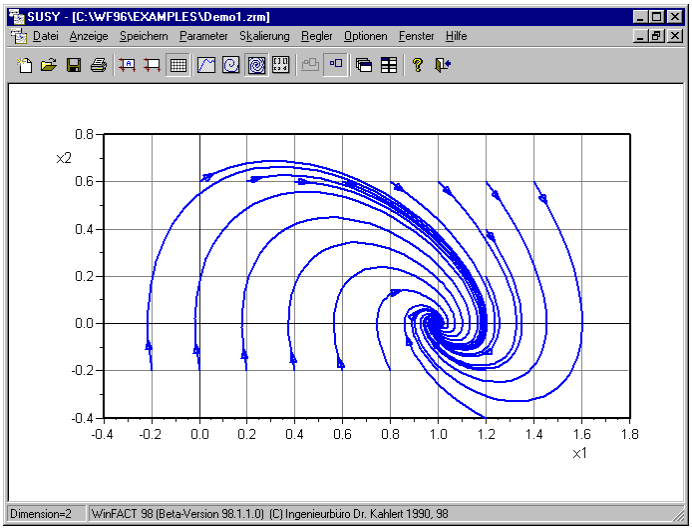
Klicken Sie den rechten Mausknopf innerhalb einer Trajektorienanzeige, so wird die Koordinate der Maus als Startposition der Trajektorie verwendet. Die Anzahl der maximal hinzufügbaren Trajektorien ist lediglich vom Arbeitsspeicher Ihres Rechners abhängig.

Hinweis: Beim Hinzufügen von Trajektorien wird der Anfangswert (System zum Zeitpunkt $t = 0$) auf die durch den Mauszeiger beschriebenen Koordinaten gesetzt.

Richtung von Trajektorien

In manchen Fällen kann es vorkommen, daß die Richtung von Trajektorien nicht klar erkennbar ist. Aus diesem Grund können Sie die Richtung der Trajektorien durch einen Pfeil kennzeichnen lassen. Ist der Menüeintrag OPTIONEN | RICHTUNG ANZEIGEN markiert, so werden die Trajektorien mit einem Pfeil

gezeichnet. Wird die Markierung durch erneutes Auswählen des Menüpunktes aufgehoben, verschwinden auch die Pfeile an den Trajektorien. Folgende Grafik zeigt ein Beispiel für ein Trajektorienfeld mit angezeigter Richtung.



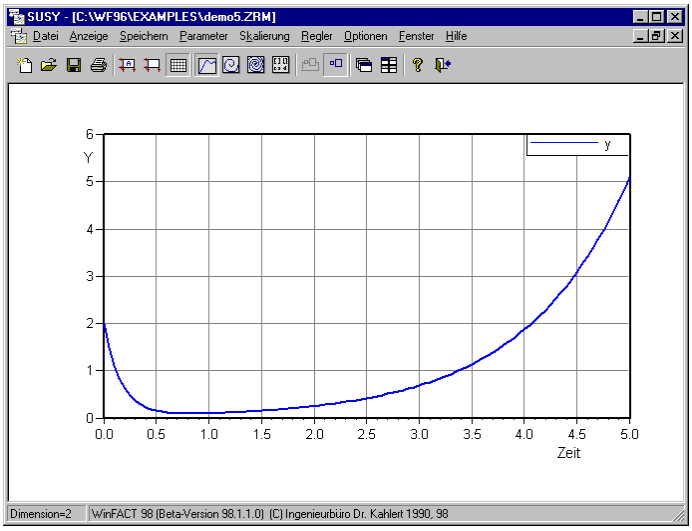
Trajektorien mit Pfeilen zur Richtungskennzeichnung

System zum Zeitpunkt Null

Der Systemzustand kann im Zeitpunkt Null von Null verschieden sein. Wollen Sie etwa Eigenschwingungen bei einer vorgegebenen Anfangsauslenkung simulieren, so setzen Sie die Eingangsfunktion zu Null und geben beim SYSTEM ZUM ZEITPUNKT '0' (unter Menüeintrag PARAMETER) die gewünschte Anfangsauslenkung ein.

Dialog zum Ändern der Systemparameter im Zeitpunkt t = 0

Bei den Anfangswerten von $x_1(0) = 1$ und $x_2(0) = 0.98$ erhalten wir für unser Beispielsystem folgenden Zeitverlauf für die Ausgangsgröße:



Verlauf der Ausgangsgröße bei einer Simulationsendzeit von 5

Reglersynthese

Es wird gezeigt, wie mit Hilfe von SUSY ein linearer Zustandsregler bestimmt werden kann. Dabei bietet SUSY Ihnen zwei halbautomatische Entwurfsverfahren sowie die Möglichkeit der manuellen Reglervorgabe. Der Vorfilter *V* zur Kompensation der bleibenden Regelabweichung wird von SUSY unabhängig von der gewählten Entwurfsmethode automatisch berechnet und bei der Simulation berücksichtigt.

Entwurf durch Polplazierung

Bei der Polplazierung geben Sie zunächst die gewünschten Eigenwerte (Pole) des geschlossenen Regelkreises durch REGLER | POLPLAZIERUNG | POLSTELLEN-EINGABE vor. Dabei hilft Ihnen der folgende Eingabedialog:

Polstellen-Eingabe zur Polplazierung

Bei der Eingabe der Polstellen sollte darauf geachtet werden, daß bei einer komplexen Polstelle auch der entsprechende konjugiert komplexe Pol angegeben werden muß. Vernachlässigen Sie dieses, so erfolgt eine Fehlermeldung.

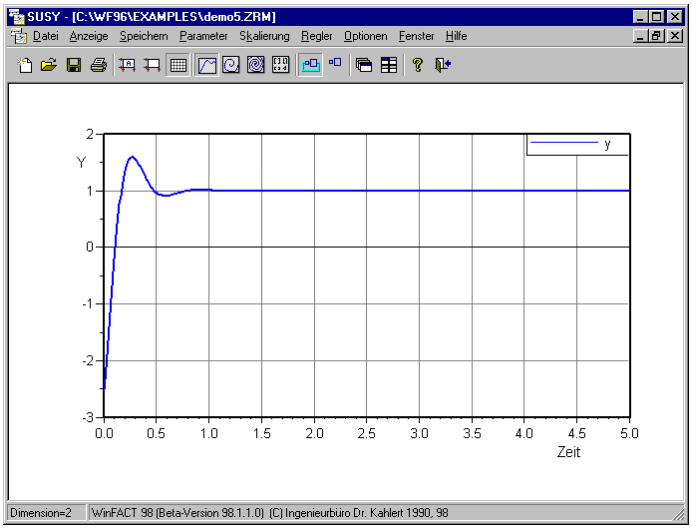
Nach Eingabe der Pole kann durch REGLER | POLPLAZIERUNG | REGLER BESTIMMEN der entsprechende Zustandsregler entworfen werden. Anschließend

kann jederzeit zwischen dem offenen und dem geschlossenen Regelkreis mit Hilfe zweier Schaltflächen in der Toolbar umgeschaltet werden.

Es soll nun ein Regler für das Beispielsystem entworfen werden, der die Pole nach $-6 \pm j10$ verschiebt. Der entworfene Regler hat folgendes Aussehen:

$$\underline{k} = \begin{pmatrix} -11.833 \\ 3.5 \end{pmatrix}.$$

Das nachfolgende Bild zeigt den Verlauf der Ausgangsgröße des geschlossenen Regelkreises bei einem Einheitssprung am Eingang.



Verlauf der Ausgangsgröße des geschlossenen Regelkreises

Riccati-Entwurf

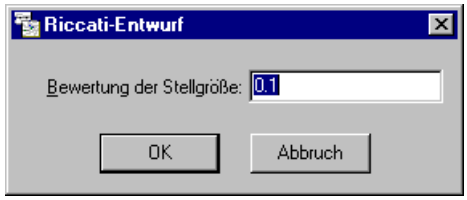
Der Riccati-Entwurf minimiert das Güteintegral

$$J = \int_0^\infty \underline{x}^T \underline{Q} \underline{x} + \beta u^2 \, dt \, .$$

Dabei sind:

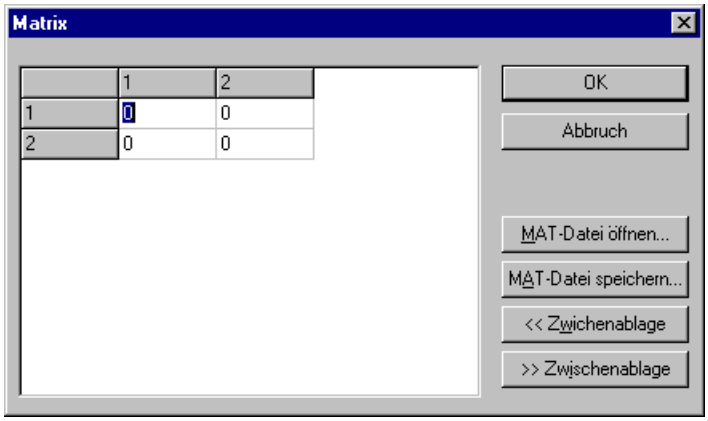
u	die Stellgröße
$\underline{x}$	der Zustandsgrößenvektor
$\underline{Q}$	die Bewertungsmatrix der Zustandsgrößen
β	die Gewichtung des Stellgrößenaufwandes

Die Bestimmung des Reglers erfolgt iterativ durch numerische Lösung der vollständigen Riccati-Gleichung. Die Parameter β und $\underline{Q}$ sind vom Anwender vorzugeben. β muß durch REGLER | RICCATI-ENTWURF | STELLGRÖßEN-BEWERTUNG festgelegt werden. Je größer β ist, desto weniger Stellaufwand wird von dem später entworfenen Regler benötigt.



Eingabe der Stellgrößenbewertung

Die Matrix $\underline{Q}$ wird durch REGLER | RICCATI-ENTWURF | MATRIX Q eingegeben. Die Dimension der Matrix ist durch die Ordnung des Systems vorgegeben und somit nicht veränderbar.

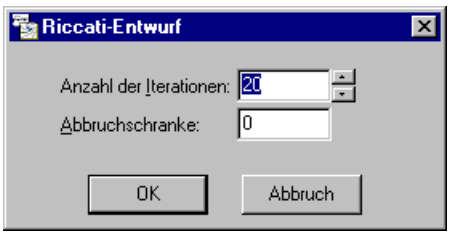


Eingabe der Bewertungsmatrix $\underline{Q}$

Als letzte, nicht im Integral auftauchende Parameter sind die zwei Entwurfsparameter

- Anzahl der Iterationen und
- Abbruchschranke

einzugeben. Sie sollten nur bei Konvergenzproblemen modifiziert werden.



Entwurfsparameter-Eingabe

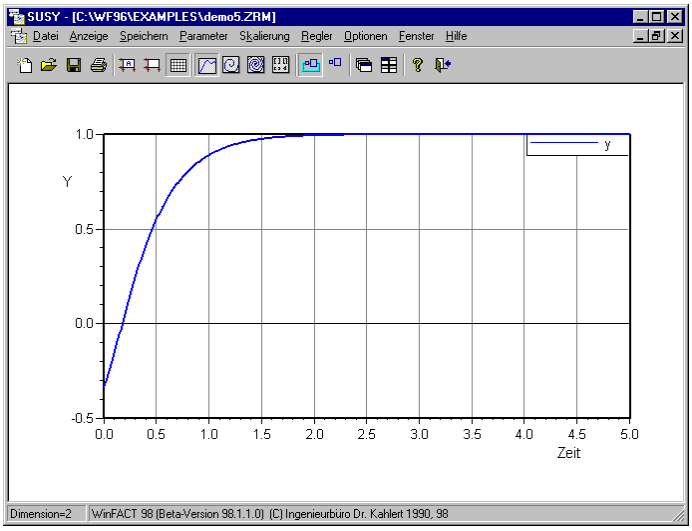
Nachfolgend soll ein Riccati-Regler für das Beispiel entworfen werden. Dabei sollen folgende Parameter eingestellt werden:

- $\underline{\underline{Q}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
- $\beta = 0.5$
- Anzahl der Iterationen: 100
- Abbruchschranke: 0.00001

Der resultierende Regler lautet

$$\underline{k} = \begin{pmatrix} -2 \\ 2 \end{pmatrix}.$$

Es ergibt sich folgender Simulationsverlauf des geschlossenen Kreises:



Verlauf der Ausgangsgröße des geschlossenen Regelkreises

Manuelle Reglereingabe

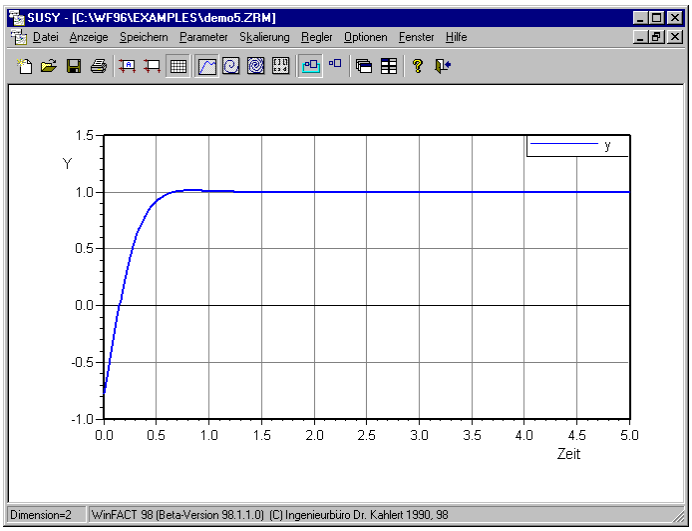
Das Menü REGLER | REGLER BEARBEITEN bietet dem Anwender die Möglichkeit, einen Regler von Hand vorzugeben bzw. nachzuoptimieren.

Manuelle Reglereingabe

Ändert man den im vorherigen Abschnitt entworfenen Regler auf

$$\underline{k} = \begin{pmatrix} -4 \\ 3 \end{pmatrix}$$

und verläßt den Dialog mit *OK*, so wird das geschlossene System automatisch mit dem neuen Regler berechnet. Die Simulation ergibt daraufhin folgendes Ergebnis:



Verwaltungsfunktionen

Möglichkeiten zur Dokumenterzeugung sowie die Steuerung der Anzeigen sind Thema dieses Abschnittes.

Textdokumente

Textdokumente können erzeugt werden, wenn Sie während der Systemanzeige den Menüeintrag SPEICHERN auswählen. Dabei werden folgende Angaben in Form einer ASCII-Datei in der von Ihnen angegebenen Datei gespeichert:

- Das offene System mit Eigenwerten
- Das geschlossene System mit Eigenwerten (falls ein Regler entworfen wurde)
- Die Parameter des Reglerentwurfs (unterschiedlich, je nachdem ob nach Riccati-Verfahren, Polplazierung oder manueller Eingabe vorgegangen wurde)
- Der Zustandsregler

Nachfolgendes Listing zeigt das Beispiel einer Dokumentdatei nach einem Riccati-Entwurf.

```
       WinFACT - Windows Fuzzy And Control Tools
          (C) Ingenieurbuero Dr. Kahlert 1998

Dokumentdatei zu: C:\tmp\test.doc
erzeugt am: 22.09.98 12:21:26

-------------------------------------------------------
Zustandsraummodell:offen

Systemmatrix A:
              0             -6
             -1             -5

Steuervektor b:
              0
              2

Ausgangsvektor c:
              2              0

Durchgriff d:
```

```
            0.5

Eigenwerte des offenen Systems:
Realteil:              Imaginaerteil:
              -6                    0
               1                    0
-----------------------------------------------------
Zustandsraummodell:geschlossen

Systemmatrix A:
               0                   -6
   2.99999952316       -8.99999904633

Steuervektor b:
               0
               2

Ausgangsvektor c:
   2.99999976158      -0.999999880791

Durchgriff d:
             0.5

Eigenwerte des geschlossenen Systems:
Realteil:              Imaginaerteil:
  -6.00000000003                    0
  -2.99999946895                    0
-----------------------------------------------------
Gütematrix Q:
               1                    0
               0                    1

Stellgrößengewichtung Beta:
0.5

Rückführungsvektor:
  -1.99999976158
   1.99999976158

Vorfilter V=  -0.666666548657
```

Speichern von Kurven

Zeitverlauf speichern

Wird der Zeitverlauf angezeigt, werden bei der Menüauswahl SPEICHERN | SPEICHERN alle angezeigten Kurven abgespeichert, und zwar in Form

- einer SIM-Datei, falls nur eine Kurve angezeigt wird,
- einer MXY-Datei, falls mehrere Kurven angezeigt werden.

Die Dateiformate sind im Kapitel 2 *Grundlagen* im Detail erläutert.

Trajektorien speichern

Sind im aktiven SUSY-Fenster Trajektorien zu sehen, so müssen Sie die abzuspeichernden Trajektorien zunächst selektieren (s. *Trajektorien auswählen*). Anschließend können Sie über SPEICHERN | SPEICHERN die selektierten Trajektorien unter gewünschtem Namen in Form einer MXY-Datei ablegen.

7 Entwurf von Fuzzy-Systemen mit der Fuzzy-Shell FLOP

Übersicht

Die Fuzzy-Shell FLOP (**F**uzzy **L**ogic **O**perating **P**rogram) ermöglicht den Entwurf und die Analyse regelbasierter Systeme basierend auf Fuzzy-Logik. Im einzelnen bietet das Programm folgende Möglichkeiten:

- Definition von linguistischen Variablen und zugehörigen Termen
- Verknüpfung und Modifikation von Fuzzy Sets
- Ermittlung von Zugehörigkeitswerten
- Erstellen von Regelwerken
- Durchführung von Inferenzvorgängen
- Ermittlung von Übertragungskennlinien und –kennfeldern
- Simulation anhand von Datensätzen
- Erstellung von Fuzzy-Controller-Dateien für das blockorientierte Simulationssystem BORIS

Für die unterschiedlichen Rechenoperationen, die i. a. grafisch dargestellt werden, stehen verschiedene Operatoren, Inferenzmechanismen und Defuzzifizierungsmethoden zur Auswahl. Für den Typ der Zugehörigkeitsfunktionen sind Dreieck, Trapez und Singleton möglich.

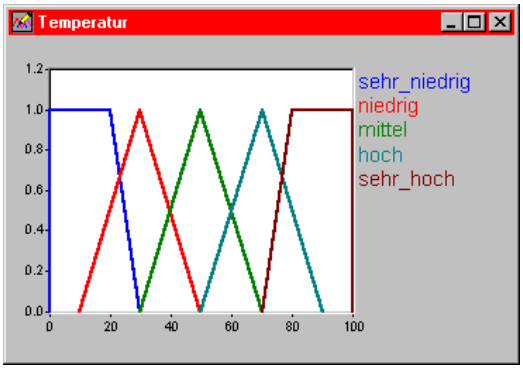
Eine komplette Einführung in die Grundlagen der Fuzzy-Logik und mögliche Anwendungsbereiche kann an dieser Stelle naturgemäß nicht erfolgen. Daher sei hier lediglich auf entsprechende Literatur (z. B. [2, 4, 9]) verwiesen. Eine hervorragende Kurzeinführung stellt [8] dar.

Die Handhabung der Fuzzy-Shell FLOP soll schrittweise anhand einfacher Beispiele erläutert werden.

Linguistische Variablen und Terme

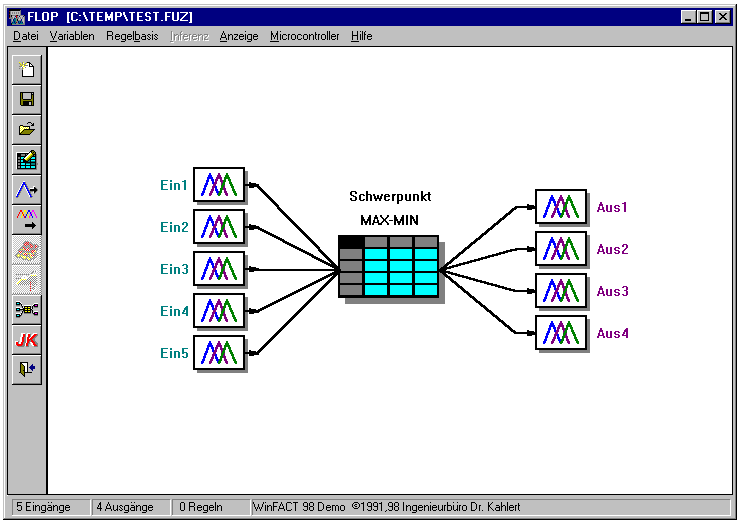
Definition und Bearbeitung

Zur Übersichtlichkeit des Programms trägt bei, daß der Benutzer jederzeit einen Überblick über die aktuell definierten linguistischen Variablen, die zugehörigen linguistischen Terme und - sofern vorhanden - die Regelbasis erhält. Jede linguistische Variable wird in einem eigenen Fenster dargestellt, das beliebig verschoben, verkleinert, vergrößert und verborgen werden kann.



Variablenfenster für eine Variable mit fünf linguistischen Termen

Die Statuszeile des Hauptfensters enthält Informationen über Anzahl und Typ der aktuell definierten linguistischen Variablen sowie die Anzahl der definierten Regeln. Im Zeichenfenster ist zudem die Struktur des augenblicklichen Fuzzy-Systems dargestellt. Um sie betrachten zu können, ist es i. a. erforderlich, zunächst alle Variablenfenster zu schließen. Dies gelingt über die Menüfolge ANZEIGE | ALLE FENSTER VERBERGEN bzw. die Systemstruktur-Schaltfläche der Toolbar. Über die Menüfolge ANZEIGE | ALLE FENSTER ANZEIGEN bzw. nochmaliges Betätigen der Toolbar-Schaltfläche werden alle Fenster wieder sichtbar. Die Anzeige der Systemstruktur kann über ANZEIGE | SYSTEMSTRUKTUR deaktiviert werden.



Anzeige der Systemstruktur für ein System mit fünf Eingängen und vier Ausgängen

Wir wollen zunächst eine linguistische Variable *Temperatur* definieren. Eine solche Variable ist festgelegt durch

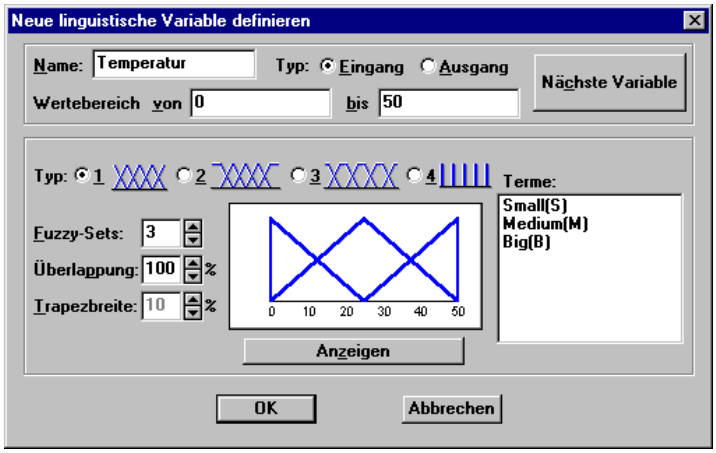
Felder einer linguistischen Variablen

- den *Typ* (Eingangs- oder Ausgangsgröße entsprechend Prämisse bzw. Konklusion der Regeln),
- ihren *Namen* (maximal 15 Zeichen),
- ihren numerischen *Wertebereich*.

Zur Definition der Variablen wählen wir die Menüoption VARIABLEN | NEUE LINGUISTISCHE VARIABLE... bzw. die entsprechende Schaltfläche der Toolbar. Wir gelangen auf diese Weise in den Dialog zur Definition und Vorbelegung neuer linguistischer Variablen. Aus diesem Dialog heraus können nacheinander mehrere neue Variablen definiert und direkt mit linguistischen Termen, also Fuzzy Sets, in verschiedenen Standardformen vorbelegt werden. Auf diese Weise ist ein sehr schnelles Erstellen der Grundstruktur des Fuzzy-Systems möglich.

Bearbeitung einer ling. Variablen

Nachfolgende Bildschirmgrafik zeigt den Eingabedialog bereits nach erfolgter Eingabe der Parameter für unsere Variable *Temperatur*. Es wurde ein Wertebereich von 0 bis 50 gewählt und dieser zunächst mit drei Fuzzy Sets in Dreiecksform mit vollständiger Überlappung der Sets vorbelegt. Die vom Programm automatisch vergebenen Bezeichnungen für die Fuzzy Sets erscheinen am rechten Rand des Dialogs in der entsprechenden Listbox.



Dialog zur Definition neuer linguistischer Variablen

Nach Neudefinition einer Variablen kann jeweils über die *Nächste Variable*-Schaltfläche die nächste Variable definiert werden, ohne daß der Dialog dazu verlassen werden muß.

Umnormierung einer Variablen

Möchten wir den Typ, den Wertebereich oder den Namen einer bereits existierenden linguistischen Variablen später ändern, so können wir dies über die Menüfolge VARIABLEN | LINGUISTISCHE VARIABLE BEARBEITEN.

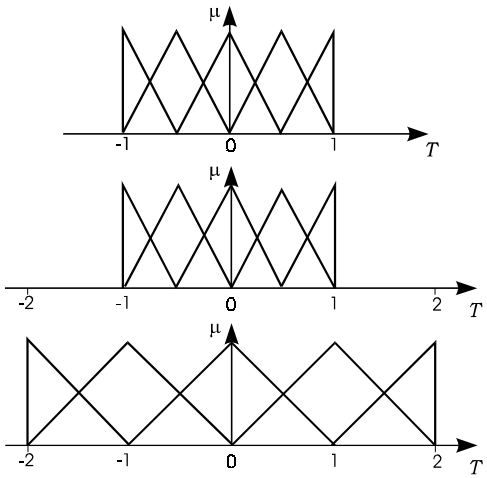
Dialog zur Bearbeitung einer linguistischen Variablen

Hier können wir nun durch Anklicken der Schaltfläche *Umbenennen* z. B. einen neuen Namen für unsere Variable eingeben. Durch das Betätigen der Schaltfläche *Löschen* können Sie eine linguistische Variable (mit allen ihren Termen)

entfernen. Wir ändern hier den Wertebereich auf Temperaturen von 0 °C bis 100 °C.

Tip: Soll das Fuzzy-System später unter BORIS benutzt werden, empfiehlt es sich, alle Ein- und Ausgangsgrößen innerhalb von FLOP auf den Wertebereich [-1, 1] bzw. [0, 1] zu normieren und die Anpassung an die tatsächlichen Wertebereiche später in BORIS durch Vor- bzw. Nachschaltung von P-Gliedern zu realisieren! Diese Vorgehensweise ist erheblich zeitsparender.

Soll der Wertebereich einer linguistischen Variablen geändert werden, die bereits Fuzzy Sets enthält, so möchte man in vielen Fällen die Fuzzy Sets selbst ebenfalls umnormieren, so daß sie den neuen Wertebereich voll ausschöpfen. Dies läßt sich über den Schalter *Fuzzy-Sets anpassen* erreichen. Wird der Schalter nicht gesetzt, so bleiben die linguistischen Terme an ihren definierten Stellen stehen. Diese Option ist sehr nützlich, wenn man später merkt, daß der Wertebereich einer linguistischen Variablen kleiner gewählt werden darf bzw. größer zu wählen ist. Durch Markierung des Auswahlfeldes *Fuzzy-Sets anpassen* werden dann die Sets dieser Variablen entsprechend des neuen Definitionsbereichs gestaucht bzw. gestreckt. Das nachfolgende Bild verdeutlicht diese Option an einem Beispiel.



Änderung des Wertebereichs einer Variablen T (oberes Bild) von [-1, 1] auf [-2, 2]. Ohne Anpassung der Fuzzy Sets ergibt sich eine Konstellation gemäß des mittleren Bildes, mit Anpassung erhält man die untere Verteilung der Fuzzy Sets.

Nach dem Verlassen des Dialogs wird der Bildschirm automatisch mit den geänderten Werten aktualisiert.

Definition neuer Fuzzy Sets

Wollen wir einer linguistischen Variablen neue Terme hinzufügen, können wir dies über die Menüfolge VARIABLEN | NEUES FUZZY-SET ... oder bequemer über die Tastenkombination <Strg><N> bzw. die Toolbar erreichen. Das nachfolgende Bild zeigt den zugehörigen Dialog nach Eingabe der ersten Fuzzy-Menge. Er legt zunächst nur die grundlegenden Parameter des Fuzzy Sets fest:

- die zugehörige linguistische Variable (hier *Temperatur*),
- den Namen des Fuzzy Sets (max. 12 Zeichen, hier *niedrig*),
- ein Kürzel für den Namen (max. 2 Zeichen, hier *N*).

Das Kürzel ist später für die übersichtliche Darstellung von Regeln in Matrixform wichtig und muß daher in jedem Fall festgelegt werden. Sollen - wie in unserem Fall - mehrere neue Terme nacheinander definiert werden, so kann dies über die Schaltfläche *Nächstes* geschehen, ohne daß dafür der Eingabedialog zwischendurch verlassen werden muß.

Dialog zur Definition neuer linguistischer Terme

Die Zugehörigkeitsfunktion selbst wird bei ihrer Definition zunächst so initialisiert, daß sie symmetrisch und dreiecksförmig ist, wobei die Einflußbreite mit dem Wertebereich der zugeordneten linguistischen Variablen identisch ist.

Alle aktuellen linguistischen Variablen werden in der Reihenfolge ihrer Definition unter der Hauptmenüoption *Anzeige* eingetragen. Zur Zeit geöffnete Variablenfenster werden durch eine Markierung gekennzeichnet (dies gilt auch für Fenster, die zum Symbol verkleinert wurden). Einmal geschlossene Variablenfenster können über diesen Menüpunkt jederzeit wieder geöffnet werden.

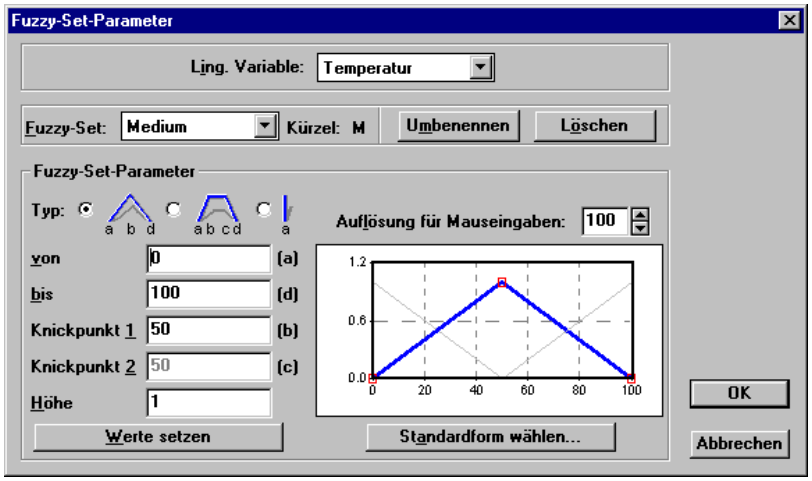
Zur Bearbeitung der Zugehörigkeitsfunktionen wechseln wir in den zentralen Eingabedialog des Programms, den wir auf verschiedene Weise erreichen können:

Bearbeiten von Fuzzy Sets

- über die Menüfolge VARIABLEN | FUZZY-SET BEARBEITEN ...,
- über die Tastenkombination<Strg><F>,
- durch einen Doppelklick mit der linken Maustaste innerhalb eines Variablenfensters. Dies ist die einfachste Möglichkeit, da in diesem Fall die angeklickte linguistische Variable automatisch im Dialog voreingestellt wird.

Von hier aus können wir folgende Aktionen durchführen:

- Bearbeiten von Zugehörigkeitsfunktionen aller linguistischen Variablen,
- Löschen und Umbenennen von linguistischen Termen.



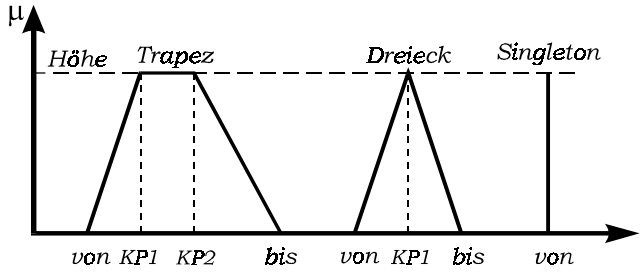
Dialog zum Bearbeiten von Fuzzy Sets

Die aktuelle linguistische Variable und die zu bearbeitende Zugehörigkeitsfunktion werden über aufklappbare Listenfenster ausgewählt. Sämtliche linguistischen Terme zur aktuellen Variablen werden grafisch im entsprechenden Fenster angezeigt. Der angewählte Term wird zusätzlich farblich und durch Markierung der charakteristischen Punkte hervorgehoben. Alle durchgeführten Änderungen werden unmittelbar protokolliert.

Wurde als Typ der Zugehörigkeitsfunktion *Singleton* gewählt, so ist lediglich die Eingabe des (scharfen) Wertes im Feld *von* notwendig. Das Eingabefeld *Knickpunkt 2* ist nur anwählbar, wenn für den Fuzzy Set-Typ die Einstellung *Trapez* gewählt wurde. Durch Eingabe einer *Höhe* kleiner als 1 lassen sich subnormale Fuzzy Sets erzeugen, mit denen allerdings im Normalfall nicht

gearbeitet werden sollte[3]. Alle Eingaben (auch Änderungen des Typs der Zugehörigkeitsfunktion) werden erst bei Betätigung der Schaltfläche *Werte setzen* übernommen.

Das nachfolgende Bild zeigt im Überblick die charakteristischen Punkte für alle Typen von Zugehörigkeitsfunktionen.



Kennwerte der einzelnen Typen von Zugehörigkeitsfunktionen

Eingabe über Maus

Alternativ zur Direkteingabe der numerischen Werte können die Fuzzy Sets auch grafisch mit der Maus editiert werden. Durch einen Mausklick mit der linken Taste auf eine Fuzzy-Menge im Anzeigefenster wird diese zunächst aktiviert. Danach können die einzelnen charakteristischen Punkte, die durch rote Quadrate markiert sind, mit der Maus bei gedrückter linker Taste verschoben werden. Das Auflösungsvermögen für den Verschiebevorgang wird durch den im Feld *Auflösung für Mauseingaben* eingestellten Wert vorgegeben. Während des Verschiebevorgangs wird das entsprechende numerische Eingabefeld automatisch aktualisiert. Eine Änderung der Höhe eines Fuzzy Sets muß jedoch in jedem Fall über das entsprechende Editierfeld vorgenommen werden.

Besondere Aufmerksamkeit verdient die Schaltfläche *Standardform*. Sie ermöglicht jederzeit eine schnelle Grundeinstellung aller Zugehörigkeitsfunktionen der angewählten linguistischen Variablen entsprechend den Möglichkeiten bei der Neudefinition der Variablen.

Anmerkung: Wird der Eingabedialog über die Schaltfläche *Abbruch* verlassen, so werden *alle* bis dahin im Eingabedialog durchgeführten Modifikationen rückgängig gemacht, d. h. der Zustand vor Aufruf des Eingabedialogs wiederhergestellt! Dies bedeutet, daß auch gelöschte oder umbenannte Fuzzy Sets wieder rekonstruiert werden.

[3] Speziell bei der C-Code-Generierung sind subnormale Sets nicht zulässig!

Für Dokumentationszwecke ist es möglich, einzelne linguistische Variablen grafisch auszudrucken oder im HPGL-Format zu exportieren. Dazu dienen die Menüfolgen VARIABLEN | LINGUISTISCHE VARIABLE DRUCKEN... bzw. VARIABLEN | LINGUISTISCHE VARIABLE EXPORTIEREN. Die auszugebende Variable kann dann über einen entsprechenden Dialog ausgewählt werden.

Berechnung von Zugehörigkeitswerten

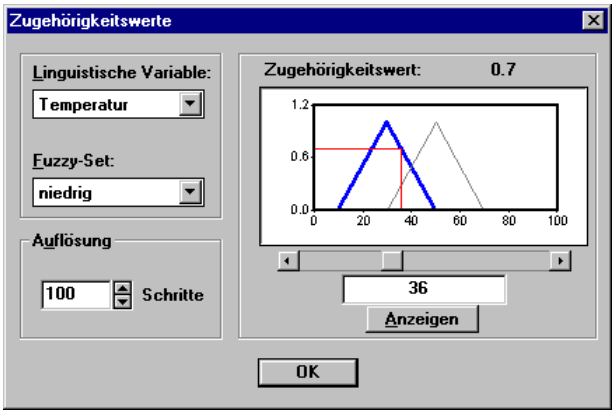
Hinweis: Die nachfolgenden beiden Abschnitte sind nur für "Einsteiger" in die Fuzzy-Logik von Interesse bzw. für denjenigen, der die Fuzzy-Shell FLOP zu Ausbildungszwecken einsetzen will. Bereits mit der Materie vertraute Anwender können diese Abschnitte ohne weiteres überspringen.

Für die nachfolgenden Untersuchungen laden wir die Datei DEMO1.FUZ aus dem WinFACT 98-Beispielverzeichnis. Diese Datei enthält eine linguistische Variable mit Namen *Temperatur* und zwei Fuzzy Sets mit Namen *niedrig* und *mittel*. Wir wollen jetzt versuchen, etwas sinnvolles mit unseren beiden Termen *niedrige Temperatur* und *mittlere Temperatur* anzufangen. Zunächst stellen wir uns die Aufgabe, für scharfe Eingangsgrößenwerte (d. h. Temperaturwerte) die Zugehörigkeit zu beiden Fuzzy Sets zu bestimmen. Dazu wählen wir die Menüfolge VARIABLEN | ZUGEHÖRIGKEITSWERTE ... oder die Tastenkombination<Strg><M>. Wir gelangen in einen Dialog, der die Berechnung von Zugehörigkeitswerten zu beliebigen Fuzzy Sets aller aktuellen linguistischen Variablen erlaubt.

Wie bereits gewohnt, können wir den gewünschten linguistischen Term über zwei aufklappbare Listenfenster anwählen. Zur Festlegung des scharfen Temperaturwertes stehen uns zwei Möglichkeiten zur Auswahl:

- Die direkte Eingabe des Wertes im Editierfeld unterhalb des Anzeigefensters und Betätigung der Schaltfläche *Anzeigen*
- Die Anwahl der Bildlaufleiste unmittelbar unter dem Anzeigefenster. Die Auflösung beträgt dabei standardmäßig 100 Schritte für den gesamten Wertebereich der linguistischen Variablen, kann aber über das Feld *Auflösung* vergrößert bzw. verkleinert werden.

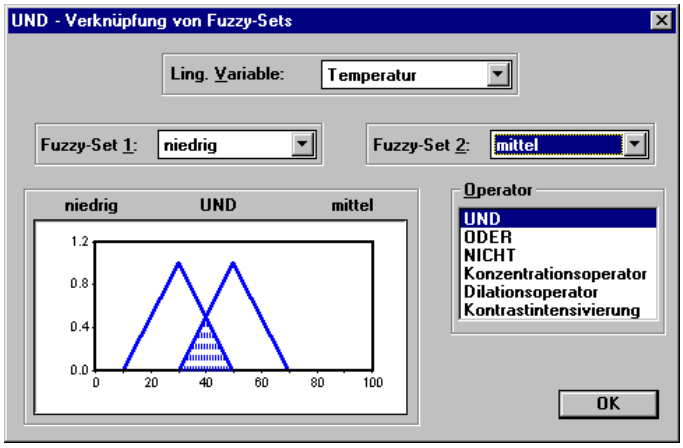
Der entsprechende Zugehörigkeitswert kann oberhalb des Anzeigefensters abgelesen werden.



Dialog zur Berechnung von Zugehörigkeitswerten

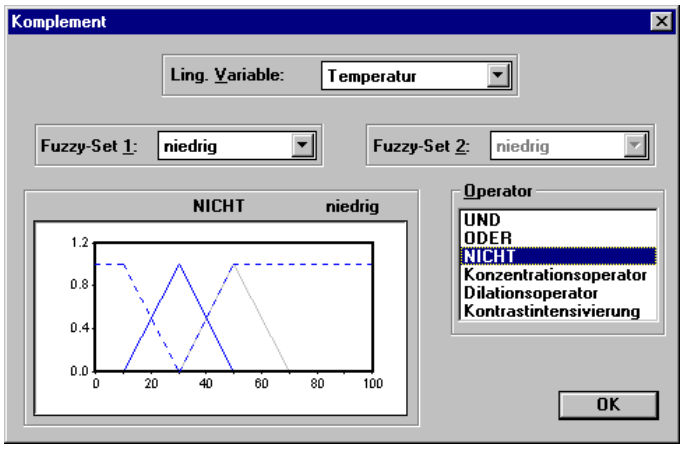
Verknüpfung und Modifikation von Fuzzy Sets

Da wir bereits zwei Fuzzy Sets für die Variable *Temperatur* definiert haben, können wir versuchen, diese miteinander zu verknüpfen bzw. zu modifizieren. Zu diesem Zweck wählen wir die Menüfolge VARIABLEN | VERKNÜPFUNGEN/MODIFIKATOREN oder die Tastenkombination <Strg><K>. Wir gelangen in den Dialog nach folgendem Bild, in dem wir über das Listenfenster am rechten Rand den Verknüpfungs- bzw. Modifikationsoperator festlegen können. Die zu verknüpfenden Terme (hier *niedrig* und *mittel*) werden über die Kombinationsfenster im oberen Teil des Dialogs ausgewählt. Die resultierende Fuzzy-Menge wird nach kurzer Zeit als schraffierte Fläche dargestellt. In unserem Fall entspricht die UND-Verknüpfung gerade dem Durchschnitt beider Fuzzy-Mengen.



Dialog zur Verknüpfung und Modifikation von Fuzzy Sets

Das Komplement eines Fuzzy Sets können wir über den NICHT-Operator ermitteln. So zeigt das nachfolgende Bild den linguistischen Term *nicht niedrig*.



Berechnung des Komplements eines Fuzzy Sets

Darüber hinaus können einzelne Zugehörigkeitsfunktionen über entsprechende Operatoren modifiziert werden. Dazu stehen folgende Modifikatoren zur Auswahl:

- Der *Konzentrationsoperator*

$$\mathrm{CON}(\mu(x)) = \mu(x)^2$$

sorgt unter Beibehaltung von Toleranz und Einflußbreite für eine Konzentration der ursprünglichen Fuzzy-Menge - die Fuzzy-Menge wird "schärfer".

- Der *Dilationsoperator*

$$\mathrm{DIL}(\mu(x)) = \sqrt{\mu(x)}$$

bewirkt den umgekehrten Effekt - die Fuzzy-Menge wird "unschärfer".

- Die *Kontrastintensivierung*

$$\mathrm{INT}(\mu(x)) = \begin{cases} 2\mu(x)^2 & \textit{für } \mu(x) < 0.5 \\ 1 - 2(1-\mu(x))^2 & \textit{sonst} \end{cases}$$

sorgt im unteren Bereich für eine Konzentration, im oberen Bereich für eine Aufspreizung der ursprünglichen Fuzzy-Menge.

Standardmäßig wird für die UND-Verknüpfung der MIN-Operator, für die ODER-Verknüpfung der MAX-Operator herangezogen. Alternativ dazu stehen jedoch weitere Operatoren zur Auswahl:

Für die UND-Verknüpfung:

- Bounded-Difference-Operator (Abgeschnittene Differenz)

$$(\mu_1 \dot{-} \mu_2)(x) := \mathrm{MAX}(0, \mu_1(x) + \mu_2(x) - 1),$$

- Algebraic-Product-Operator (Algebraisches Produkt)

$$(\mu_1 \mu_2)(x) := \mu_1(x) \cdot \mu_2(x).$$

Für die ODER-Verknüpfung:

- Bounded-Sum-Operator (Abgeschnittene Summe)

$$(\mu_1 \hat{+} \mu_2)(x) := \mathrm{MIN}(1, \mu_1(x) + \mu_2(x)),$$

- Algebraic-Sum-Operator (Algebraische Summe)

$$(\mu_1 \oplus \mu_2)(x) := \mu_1(x) + \mu_2(x) - \mu_1(x) \cdot \mu_2(x).$$

Diese können über VARIABLEN | OPERATOREN oder <Strg><O> erreicht werden.

Dateioperationen

Alle relevanten Daten werden in Dateien mit der Extension FUZ abgelegt. Diese enthalten

- alle linguistischen Variablen mit den zugehörigen linguistischen Termen,
- die aktuelle Regelbasis (sofern vorhanden).

Der genaue Aufbau einer FUZ-Datei ist am Ende dieses Kapitels erläutert.

Alle sonstigen Einstellungen (z. B. die Wahl der Verknüpfungsoperatoren) werden dagegen nicht abgespeichert. Der Name der Datei wird nach dem Abspeichern in der Titelzeile des Hauptfensters angezeigt.

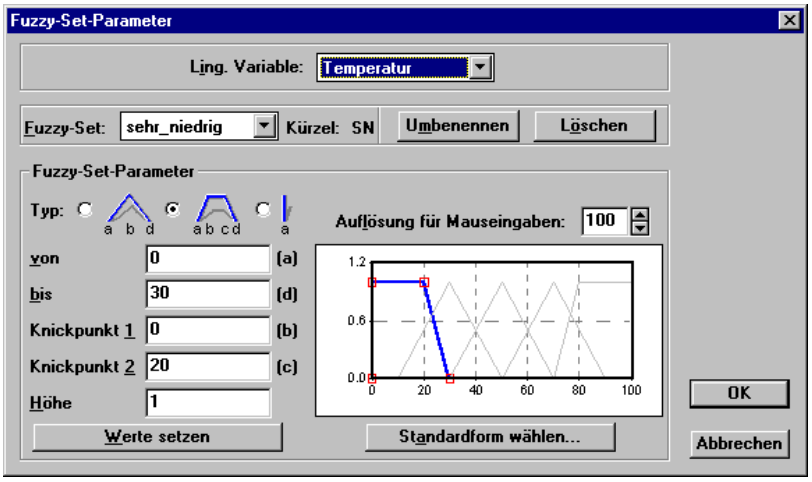
Definition und Bearbeitung einer Regelbasis

Wir wollen uns im folgenden mit der Erstellung und Bearbeitung einer Regelbasis beschäftigen. Dabei werden wir uns zunächst auf den Fall *einer* Prämisse beschränken. Eingangsgröße unserer Regelbasis (WENN-Teil der Regeln) soll die bereits vorliegende Variable *Temperatur* sein. Da wir bisher lediglich zwei Terme, nämlich *niedrig* und *mittel* definiert haben, wollen wir die Variable durch die Terme *sehr_niedrig*, *hoch* und *sehr_hoch* vervollständigen. Die entsprechenden Zugehörigkeitsfunktionen sollen gemäß nachfolgendem Bild gewählt werden.[1] Wir erkennen, daß für die Randmengen trapezförmige Fuzzy Sets gewählt wurden.

Im Listenfenster der Fuzzy Sets werden die einzelnen Terme zunächst in der Reihenfolge angezeigt, in der sie über VARIABLEN | NEUES FUZZY-SET definiert

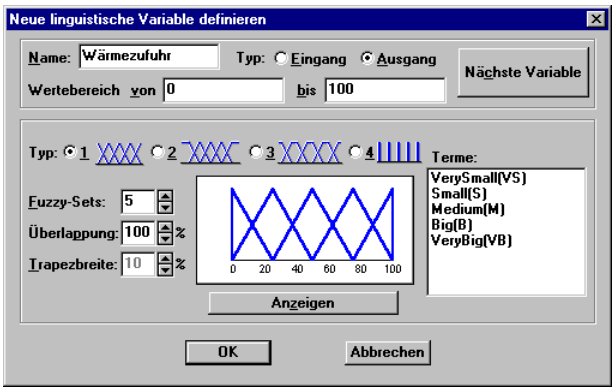
[1] Der komplette Datensatz (allerdings noch ohne Regelbasis) befindet sich unter dem Namen DEMO2.FUZ im Beispiel-Verzeichnis.

wurden. Nach dem Verlassen des Eingabedialogs werden sie vom Programm automatisch sortiert und in der korrekten Reihenfolge angezeigt.



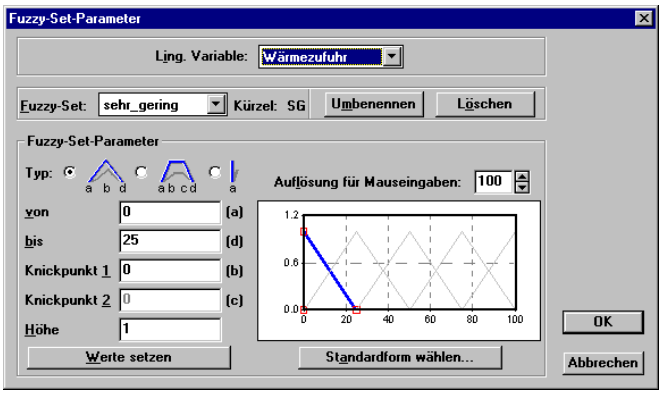
Linguistische Variable Temperatur

Wir wollen als Problemstellung das Erhitzen von Wasser betrachten. Die linguistische Variable im DANN-Teil unserer Regeln ist daher die *Wärmezufuhr*, die den Wertebereich von 0 bis 100 % überstreichen soll. Zunächst wählen wir also die Option VARIABLEN | NEUE LINGUISTISCHE VARIABLE ... bzw. <Strg><V> und geben die entsprechenden Daten an.



Definition der linguistischen Variablen Wärmezufuhr

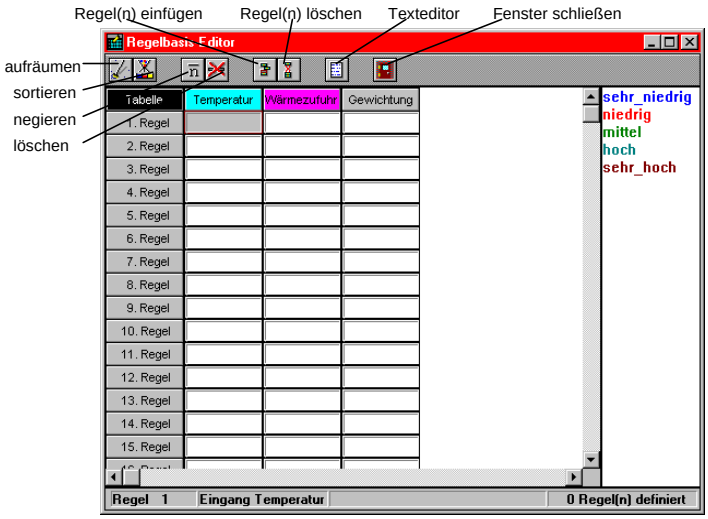
Für diese Variable - die wir als Ausgangsgröße zu definieren haben - sollen die Terme *sehr_gering*, *gering*, *mittel*, *stark* und *sehr_stark* mit den Kürzeln *SG*, *G*, *M*, *S* und *SS* definiert werden. Die entsprechende Umbenennung der voreingestellten Bezeichner können wir im Fuzzy Set-Parameter-Dialog vornehmen. Nachfolgender Dialog zeigt die entsprechenden Zugehörigkeitsfunktionen.



Linguistische Variable Wärmezufuhr

Über die Menüfolge REGELBASIS | NEU ... gelangen wir in das Eingabefenster für die Regelbasis. Dieses Fenster kann - genauso wie die Variablenfenster - jederzeit verschoben, verkleinert, vergrößert und verborgen werden. Ebenso wird es unter der Menüoption *Anzeige* in die Liste der aktuellen Fenster eingetragen. Nachfolgendes Bild zeigt den Regelbasis-Editor unmittelbar nach dem erstmaligen Aufruf. Wir erkennen, daß die Regelbasis in Tabellenform aufgebaut ist, wobei jede Zeile der Tabelle einer Regel entspricht. Spalten mit Eingangsgrößen sind türkis, Spalten mit Ausgangsgrößen violett markiert.

Die Definition von Regeln wird vollständig mausgesteuert vorgenommen. Dazu enthält das Listenfenster am rechten Fensterrand alle linguistischen Terme der markierten Variablen. Durch Doppelklick mit der linken Maustaste wird der angewählte Term in die Regelbasis übernommen. Die Statuszeile zeigt den markierten Eintrag (Nummer der Regel, linguistische Variable und die Gewichtung der Regel) noch einmal an und gibt weiterhin einen Überblick über die Gesamtzahl definierter Regeln.



Regelbasis-Editor nach dem Aufruf

Die *Aufräumen*-Schaltfläche ermöglicht das Entfernen ungültiger Regeln aus der Regelbasis. Als ungültig gelten solche Regeln, bei denen nicht bei mindestens einer Ausgangsgröße ein Eintrag vorhanden ist. Dieser Aufräumvorgang erfolgt in jedem Falle automatisch vor dem Abspeichern einer Datei. Über die *Sortieren*- Schaltfläche können die Regeln darüberhinaus nach linguistischen Termen in den jeweiligen Prämissen sortiert werden. Durch Anklicken der *Negieren*- Schaltfläche wird eine Teilprämisse, falls vorhanden, negiert. Die Regel enthielte somit in einem Teil der Prämisse den NICHT - Operator, z. B. WENN *Temperatur* = NICHT *niedrig* ... Weiterhin können ganze Regeln eingefügt und gelöscht werden (Schaltflächen *Regel(n) einfügen* und *Regel(n) löschen*). Ein komfortabler *Texteditor*, auf den später eingegangen wird, bietet die Möglichkeit, Regeln direkt in Textform einzugeben oder zu laden.

Wir wollen nun folgende Regeln definieren:

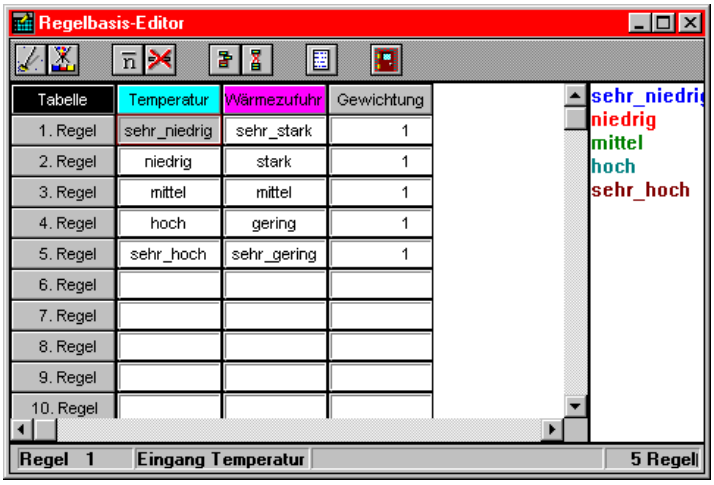
WENN	*Temperatur = sehr_niedrig*	DANN	*Wärmezufuhr = sehr_stark*
WENN	*Temperatur = niedrig*	DANN	*Wärmezufuhr = stark*
WENN	*Temperatur = mittel*	DANN	*Wärmezufuhr = mittel*
WENN	*Temperatur = hoch*	DANN	*Wärmezufuhr = gering*
WENN	*Temperatur = sehr_hoch*	DANN	*Wärmezufuhr = sehr_gering*

Die erste Regel entspricht der ersten Zeile der Regelbasis, die zweite Regel der zweiten Zeile usw. Folgendes Bild zeigt den Regelbasis-Editor nach Festlegung

sämtlicher Regeln. Bei mehreren Eingangsgrößen sind die entsprechenden Prämissen automatisch UND-verknüpft.

Hätten wir die Regeln in einer anderen Reihenfolge definiert, könnten wir durch *Sortieren* die oben vorgegebene Reihenfolge erhalten. Die Gewichtung einer Regel wird voreingestellt automatisch auf 1 gesetzt.

Soll die Gewichtung einer Regel geändert werden (in unserem Beispiel ist das allerdings nicht der Fall), so wird der entsprechende Eintrag innerhalb der Regelbasis angewählt. Es erscheint daraufhin am rechten Rand ein Scrollbalken, über den der gewünschte Wert eingestellt werden kann.



Regelbasis-Editor nach Eingabe der Regeln

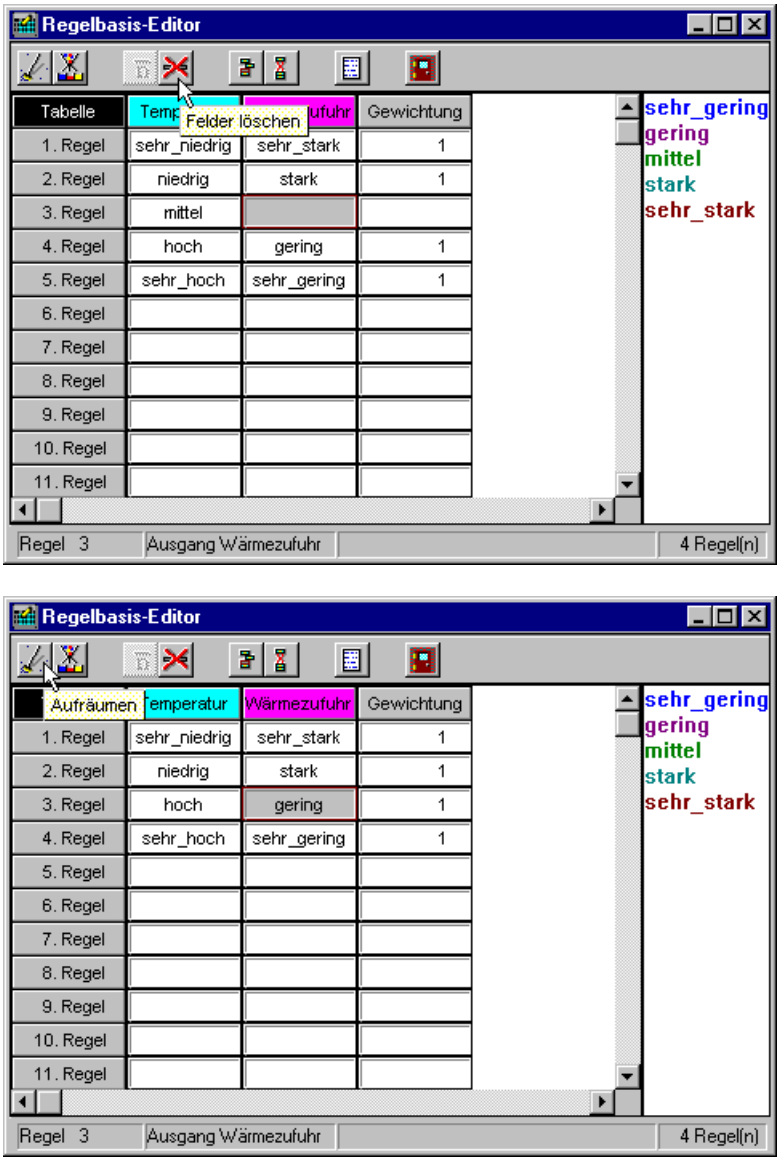
Ändern der Gewichtung einer Regel

Die *Schließen*- Schaltfläche erlaubt ein temporäres Schließen, d. h. Verbergen des Fensters. Es wird beim nächsten Aufruf der Menüoption REGELBASIS | BEARBEITEN automatisch wieder angezeigt.

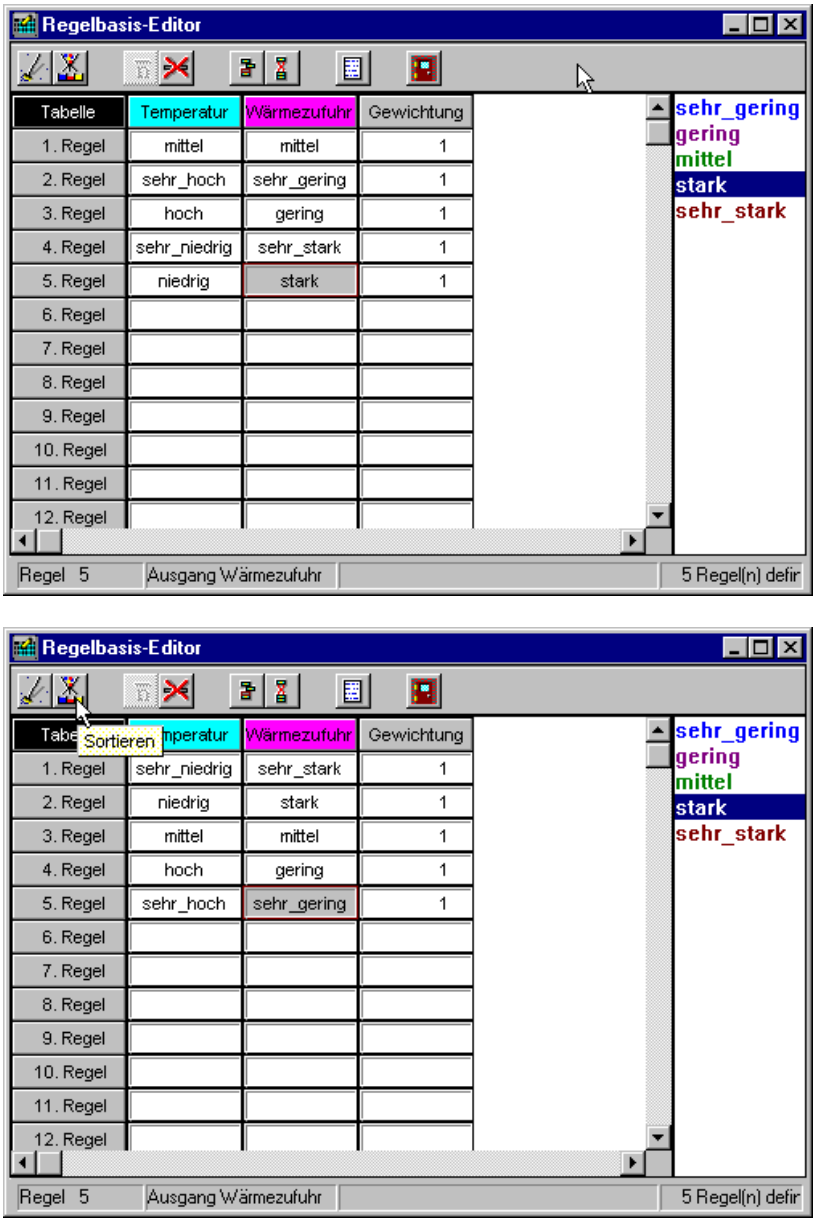
Neben der Möglichkeit, einzelne Einträge der Regelbasis zu setzen oder zu löschen, bietet der Regelbasis-Editor weitergehende Funktionen, die insbesondere bei komplexeren Systemen hilfreich sind und eine komfortable und schnelle Erstellung und Modifikation der Regelbasis gestatten:

- Durch Festhalten der <Shift>-Taste beim Anklicken von Einträgen können mehrere untereinanderliegende Einträge einer Spalte gleichzeitig gewählt werden.
- Durch Festhalten der <Strg>-Taste können mehrere, nicht zwangsläufig direkt untereinander liegende Einträge einer Spalte gleichzeitig angewählt werden.

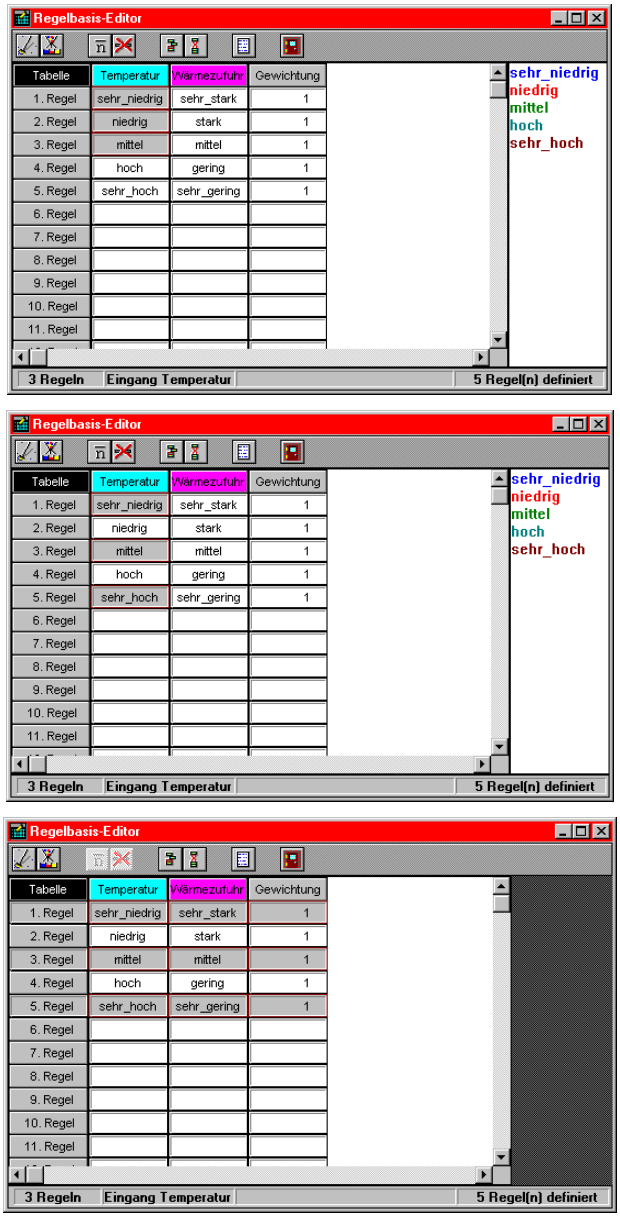
Die derart selektierten Felder können dann gleichzeitig gelöscht, mit demselben linguistischen Term gefüllt, bzw. mit derselben Gewichtung, falls diese Spalte die Markierungen enthält, versehen oder an eine andere Stelle der entsprechenden Spalte kopiert werden. Der Kopiervorgang wird dadurch eingeleitet, daß Sie den linken Mausknopf auf einer selektierten Fläche drücken und **nicht** loslassen! Nun ziehen Sie die Maus an die entsprechende Stelle, an der Sie die Kopie einfügen wollen und lassen den Mausknopf los. Dabei hilft FLOP Ihnen dadurch, daß der Cursor sich beim Ziehen verändert. Ist das Einfügen an einer Stelle zulässig, so nimmt er die Form eines Zeigers an, sonst die Form eines Kreises mit einer schrägen Linie (Halteverbotschild). Ebenfalls können ganze Regeln ausgewählt werden. Dazu sind im Regelbasiseditor die entsprechenden Felder der ersten Spalte anzuwählen. Die nachfolgenden Bilder erläutern noch einmal die unterschiedlichen Optionen des Regelbasis-Editors.



Löschen von Regeln: Löschen der Konklusion (oben) und Betätigung der Schaltfläche "Aufräumen" (unten)



Sortieren von Regeln: Regeln nach der Eingabe (oben) und nach Betätigung der Schaltfläche "Sortieren" (unten)



Gleichzeitige Selektierung mehrerer Einträge der Regelbasis

Inferenz

Einzelschritt-Inferenz

Inferenz (Einzelschritt)

Nach der Definition der Regelbasis können wir feststellen, daß das Untermenü INFERENZ im Hauptmenü nunmehr aktivierbar ist. Wählen wir hier die Option EINZELSCHRITT an, so gelangen wir in den Inferenzdialog. Er ermöglicht es uns, zu scharfen Werten der Eingangsgröße *Temperatur* über den Inferenzmechanismus und die Defuzzifizierung scharfe Werte der Ausgangsgröße *Wärmezufuhr* zu berechnen.

Die Hauptkomponenten des Inferenzdialogs sind:

- Der Bereich für die Einstellung konstanter Werte (linker Fensterrand)

  Es können jeweils nur maximal zwei Eingangsgrößen variiert werden. Liegen mehr als zwei Eingangsgrößen vor, so sind für die restlichen Eingangsgrößen die betreffenden festen Werte anzugeben, für die der Inferenzvorgang durchgeführt werden soll. Dazu wird die entsprechende Variable im Listenfenster ausgewählt, der gewünschte Wert im darunterliegenden Editierfeld vorgegeben und durch Anwahl der Schaltfläche *Setzen* übernommen. Bei nur einer oder zwei Eingangsgrößen wird dieser Teil des Dialogs naturgemäß nicht benötigt und bleibt daher passiv.

- Die Anzeigefenster für die Eingangsgröße (in der Mitte des Dialogs)

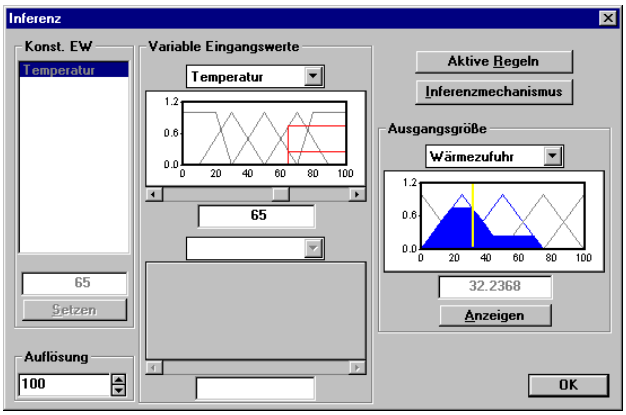
  Die interaktiv zu variierenden Eingangsgrößen können über die entsprechenden Kombinationsfenster ausgewählt werden. Der jeweils aktuelle Wert kann, wie bei der Berechnung von Zugehörigkeitswerten, entweder direkt unterhalb des Fensters eingegeben und durch Betätigen der Schaltfläche *Anzeigen* übernommen oder über die Bildlaufleiste variiert werden. Die Auflösung des Scrollvorgangs ist einstellbar. Bei nur einer Eingangsgröße - wie in diesem Beispiel - bleibt das untere Fenster leer.

- Das Anzeigefenster für die Ausgangsgröße (rechts)

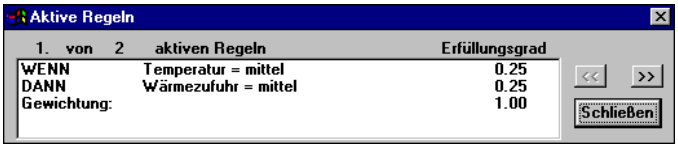
  Dieses Fenster zeigt die resultierende Ausgangsgrößen-Fuzzy-Menge und den ermittelten scharfen Ausgangsgrößenwert grafisch an. Letzterer wird zusätzlich unterhalb des Fensters numerisch angezeigt. Bei mehr als einer Ausgangsgröße kann die anzuzeigende Größe über das aufklappbare Listenfenster oberhalb des Anzeigefensters ausgewählt werden.

Anzeige aktiver Regeln

Die jeweils aktiven Regeln und ihr Erfüllungsgrad können über die Schaltfläche *Aktive Regeln* abgerufen werden. Sie werden dann in einem nichtmodalen Fenster oberhalb des Dialogs angezeigt. Dieses Fenster kann beliebig verschoben werden und wird, solange es sichtbar ist, ständig aktualisiert. Bei mehreren aktiven Regeln kann über die Schaltflächen >> bzw. << innerhalb der Regeln geblättert werden.



Inferenzdialog bei einer Eingangsgröße



Anzeige aktiver Regeln

Über die Schaltfläche *Inferenzmechanismus* gelangen wir in einen Dialog, der auch über das Hauptmenü mittels INFERENZ | INFERENZMECHANISMUS U. DEFUZZIFIZIERUNG erreicht werden kann. Über diesen Dialog kann zwischen MAX-MIN-Inferenz und MAX-PROD-Inferenz gewählt werden. Weiterhin

stehen in Form der Schaltergruppe *Defuzzifizierung* verschiedene Defuzzifizierungsmethoden zur Auswahl, die in [2, 4] im Detail erläutert werden:

- Schwerpunktmethode (Centre of Gravity - Method)

  Bei der (originalen) Schwerpunktmethode wird der Abszissenwert des Flächenschwerpunktes S der resultierenden Ausgangs-Fuzzy-Menge $\mu_{res}(y)$ zur scharfen Ausgangsgröße y_{res} gewählt. Die exakte Formel lautet

$$y_{res} = \frac{\int_0^\infty y\mu_{res}(y)\mathrm{d}y}{\int_0^\infty \mu_{res}(y)\mathrm{d}y}.$$

  Das Integral wird programmintern durch numerische Integration mit vorgebbarer Stützstellenzahl ausgewertet.

- Modifizierte (randerweiterte) Schwerpunktmethode

  Diese Methode entspricht im Prinzip der originalen Schwerpunktmethode, die beiden Randmengen der Ausgangsgröße werden jedoch symmetrisch erweitert, damit der minimal bzw. maximal mögliche scharfe Ausgangswert gerade mit dem Wertebereich der Ausgangsgröße übereinstimmt. Das Ergebnis unterscheidet sich von der originalen Schwerpunktmethode also nur dann, wenn eine der Randmengen einen Beitrag zur resultierenden Ausgangs-Fuzzy-Menge liefert.

- Näherungsweise Schwerpunktmethode (Höhenmethode, Schwerpunktmethode für Singletons)

  Da die originale Schwerpunktmethode sehr rechenzeitaufwendig ist, empfiehlt sich i. a. die Anwendung einer Näherungsformel für den Schwerpunkt. Diese lautet

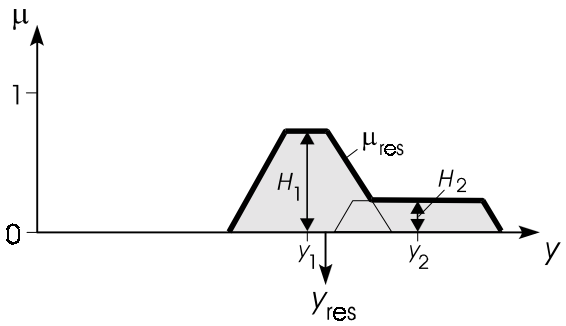
$$y_{res} \approx \frac{\sum_{i=1}^{m} y_i H_i}{\sum_{i=1}^{m} H_i}.$$

Darin ist:

m:	die Anzahl der Regeln
H_i:	der Erfüllungsgrad der i-ten Regel
y_i:	der Abszissenwert des Schwerpunktes der i-ten Ausgangsmenge

Für untenstehende Ausgangs-Fuzzy-Menge erhält man also beispielsweise die Beziehung

$$y_{res} \approx \frac{y_1 H_1 + y_2 H_2}{H_1 + H_2} .$$



Näherungsformel für Schwerpunktmethode

Sind für die Ausgangsgröße y Singletons definiert, so stimmt die Näherungsformel mit der exakten Gleichung überein.

- Defuzzifizierung nach maximaler Höhe (Maximum-Methode)

  Es wird nur die Regel mit dem maximalen Erfüllungsgrad betrachtet. Der Modalwert der zugehörigen Ergebnis-Fuzzy-Menge liefert die scharfe Ausgangsgröße. Besitzen mehrere Regeln gleichzeitig den maximalen Erfüllungsgrad, so kann die am weitesten links oder rechts liegende Ausgangs-Fuzzy-Menge gewählt werden. Man spricht daher von der Maximum-links- bzw. Maximum-rechts-Methode.

- First of Maxima-Methode

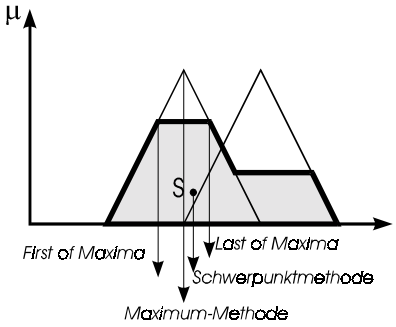
  Es wird nur die Regel mit dem maximalen Erfüllungsgrad betrachtet und die zugehörige Ergebnis-Fuzzy-Menge in der Höhe des Erfül-

lungsgrades abgeschnitten. Der linke Knickpunkt der so entstehenden Fuzzy-Menge liefert die scharfe Ausgangsgröße [4].

- Last of Maxima-Methode

  Es wird nur die Regel mit dem maximalen Erfüllungsgrad betrachtet und die zugehörige Ergebnis-Fuzzy-Menge in der Höhe des Erfüllungsgrades abgeschnitten. Der rechte Knickpunkt der so entstehenden Fuzzy-Menge liefert die scharfe Ausgangsgröße [4].

Die nachfolgende Grafik verdeutlicht die unterschiedlichen Defuzzifizierungsmethoden anhand eines Beispiels.



Defuzzifizierungsverfahren

Außerdem kann über den Dialog das Verhalten des Systems für den Fall festgelegt werden, daß keine der definierten Regeln aktiv ist. Es bestehen prinzipiell zwei Möglichkeiten:

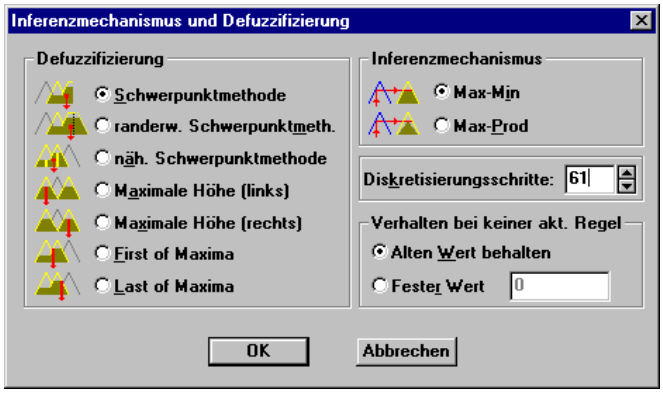
Verhalten bei keiner aktiven Regel

- Der letzte gültige Ausgangswert wird beibehalten.
- Es wird ein fester Vorgabewert ausgegeben. Dieser kann im entsprechenden Eingabefeld vorgegeben werden.

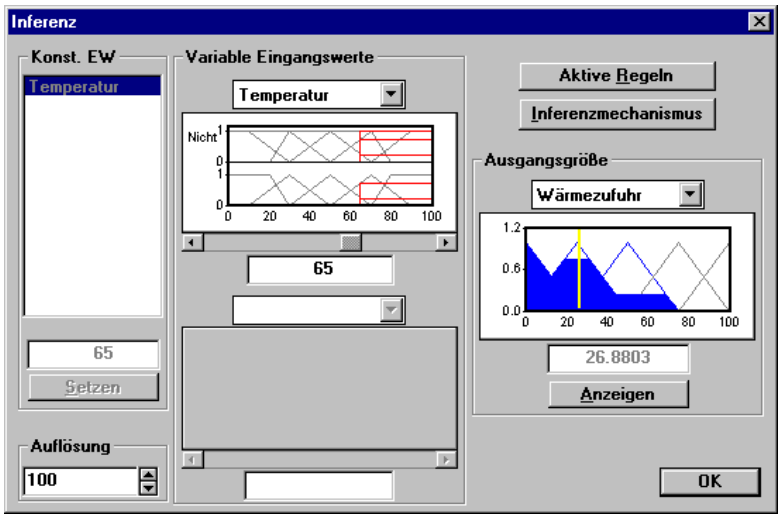
Der Parameter *Diskretisierungsschritte* schließlich legt fest, wieviele Stützstellen bei der Schwerpunktmethode zur Annäherung des Integrals benutzt werden. Voreingestellt ist ein Wert von 61.

Beim Wechsel von Inferenzmechanismus oder Defuzzifizierungsmethode werden die Anzeigen automatisch aktualisiert.

Ist eine Teilprämisse einer Eingangsgröße negiert (z. B. WENN *Temperatur* = NICHT *niedrig* ...), so erscheinen im entsprechenden Anzeigefenster im Einzelschrittdialog zusätzlich zu den Zugehörigkeitsfunktionen der linguistischen Terme selbst auch die jeweils komplementären Zugehörigkeitsfunktionen.



Dialog zur Wahl von Inferenzmechanismus und Defuzzifizierungsart



Inferenzdialog bei negierten Teilprämissen

Kennlinien- und Kennfeldberechnung

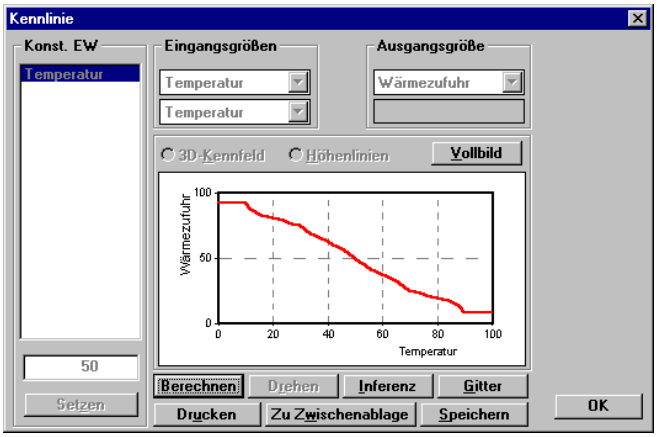
Das Übertragungsverhalten eines Fuzzy-Systems kann als Kennlinie bzw. Kennfeld dargestellt werden. Dazu dient die Menüoption INFERENZ | KENNLINIE/KENNFELD. Im Falle nur einer Eingangsgröße wird die entsprechende Kennlinie berechnet, bei zwei Eingangsgrößen das Kennfeld. Liegen mehr als

Kennlinie Kennfeld

zwei Eingangsgrößen vor, so sind die zu variierenden Größen über die Eingangsgrößen-Listenfenster auszuwählen, während für die anderen Eingangsgrößen konstante Werte (vgl. Einzelschrittdialog) vorzugeben sind.

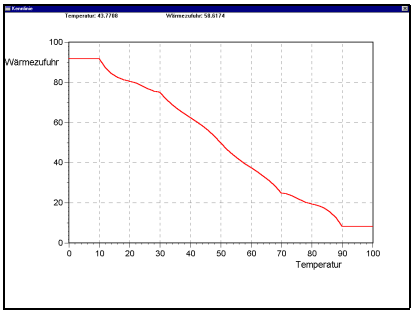
Der Dialog bietet zusätzlich folgende Schaltflächen:

- Über *Drucken* kann die Kennlinie bzw. das Kennfeld auf dem momentan installierten Drucker ausgedruckt werden.
- Über *Zu Zwischenablage* kann das Kennfeld als Bitmap in die Zwischenablage kopiert werden.
- Eine Abspeicherung der Kennlinie in einer Datei vom Typ XY ist über *Speichern* möglich.
- Die Anzahl der Stützpunkte für die Kennlinienberechnung kann über *Gitter* gewählt werden. Voreingestellt ist ein Wert von 51. Erlaubt sind Werte zwischen 11 und 201.



Kennlinien-/Kennfelddialog bei einer Eingangsgröße

- Ein Ändern des Inferenzmechanismus und der Defuzzifizierungsmethode ist über *Inferenz* möglich.
- Eine *Vollbild* -Darstellung zur genaueren Betrachtung der Kennlinie ist ebenfalls möglich. Dabei werden die Mauskoordinaten, sofern diese innerhalb des gezeichneten Koordinatensystems liegen, als Eingangswert herangezogen und so der entsprechende Ausgangswert genau bestimmt. Sollte Ihre Kurve also von den oberhalb des Kennlinienfeldes angegebenen Werten abweichen, liegt das an der Auflösung der Kennlinie!



Kennlinie in Vollbilddarstellung

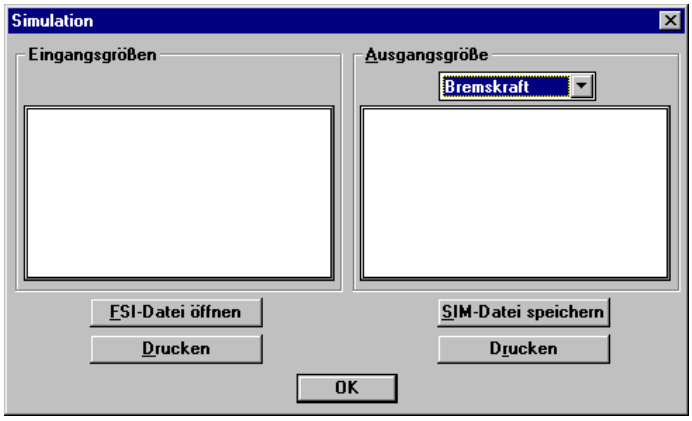
Simulation

Alternativ zum Einzelschritt- bzw. Kennfeldmodus kann das Übertragungsverhalten eines Fuzzy-Systems für beliebige Verläufe der Eingangsgröße(n) über die Menüfolge INFERENZ | SIMULATION ermittelt werden. In diesem Fall werden die Eingangswerte aus einer Datei mit der Extension FSI gelesen und der resultierende Ausgangsgrößenverlauf grafisch angezeigt. Nachfolgende Grafik zeigt den zugehörigen Dialog vor dem Einlesen der Simulationsdaten. Der Dialog ermöglicht:

- Das Einlesen der Eingangswerte (Schaltfläche ganz links). Die eingelesenen Werte werden im linken Anzeigefenster protokolliert. Um alle Eingangsverläufe mit der gleichen Skalierung darstellen zu können, werden diese für die Anzeige jeweils auf ihren Maximalwert normiert, so daß alle angezeigten Werte zwischen -1 und +1 liegen.
- Das Ausdrucken der Fensterinhalte.
- Das Abspeichern des Ausgangsgrößenverlaufs in einer SIM-Datei.

Die Eingabedatei muß in jeder Zeile die Eingangswerte für einen Zeitpunkt enthalten, wobei die Einzelwerte durch ein oder mehrere Leerzeichen zu trennen sind. Eine Zeitparametrierung findet nicht statt, so daß nur die Eingangswerte selbst anzugeben sind. Bei einem System mit zwei Eingangsgrößen enthält jede Zeile somit zwei Elemente. Nachfolgende Tabelle zeigt einen Beispieldatensatz für ein System mit zwei Eingangsgrößen.

In diesem Fall wird das Übertragungsverhalten des Fuzzy-Systems für den Fall simuliert, daß die erste Eingangsgröße linear von 1 auf 5 ansteigt, während die zweite Eingangsgröße einen konstanten Wert von 10 aufweist. Ein Anwendungsbeispiel folgt im nachfolgenden Abschnitt.



Dialog zur Simulation (vor dem Einlesen der Simulationsdaten)

```
1           10
2           10
3           10
4           10
5           10
```

Beispiel für eine FSI-Simulationsdatei

Fuzzy-Systeme mit mehreren Eingangsgrößen

In diesem Abschnitt wollen wir ein etwas komplexeres Problem betrachten, das sich vom vorangegangenen Beispiel im wesentlichen dadurch unterscheidet, daß zwei Eingangsgrößen vorhanden sind, der WENN-Teil der Regeln also zwei Prämissen enthält. Wir wollen uns vorstellen, wir befänden uns mit unserem Wagen auf der Autobahn und hätten die Aufgabe, abhängig von

- dem *Abstand* unseres Wagens zum vorausfahrenden Fahrzeug,
- der *Geschwindigkeit* unseres Wagens

einen Bremsvorgang auszuführen, d. h.

- die *Bremskraft*

vorzugeben.

Dazu definieren wir für unsere linguistischen Variablen zunächst folgende Terme (Kürzel in Klammern):

- für den Abstand und die Geschwindigkeit die Terme *sehr_niedrig* (*SN*), *niedrig* (*N*), *mittel* (*M*), *hoch* (*H*) und *sehr_hoch* (*SH*),
- für die Bremskraft die Terme *null* (*NU*), *viertel* (*VI*), *halb* (*HA*), *dreiviertel* (*DV*) und *voll* (*VO*).

Auf die Definition der Zugehörigkeitsfunktionen für die einzelnen Terme wollen wir an dieser Stelle verzichten, da wir sämtliche Daten später aus einer Datei lesen wollen. Vielmehr wollen wir uns der Regelbasis zuwenden. Bei zwei Eingangsgrößen und einer Ausgangsgröße kann diese nämlich alternativ zur Tabellenform in Matrixform dargestellt werden. Allerdings gilt hier die Bedingung, daß keine der Teilprämissen negiert sein darf und jede Teilprämisse mit einem linguistischen Term besetzt sein muß!

Regelbasis in Matrixform

Regeln der Art

WENN *Abstand = niedrig* DANN *Bremskraft = dreiviertel*

oder

WENN *Abstand = niedrig* UND *Geschwindigkeit* = NICHT *mittel*
DANN *Bremskraft = dreiviertel*

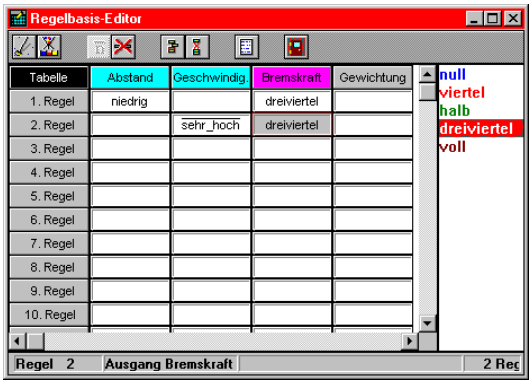
lassen sich in der Matrixform also nicht darstellen. Daher ist ein Wechsel zu dieser in solchen Fällen nicht möglich. Dennoch können Regeln dieser Form durchaus sinnvoll sein. Es ist also nicht zwangsläufig notwendig, bei einer Regel alle Prämissen anzugeben. Wollte man beispielsweise die Regel

WENN *Abstand = niedrig*
ODER *Geschwindigkeit* = NICHT *sehr_hoch*
DANN *Bremskraft = dreiviertel*

eingeben, so muß dieses innerhalb der Tabellenform oder der Textform (s. u.) geschehen. Die Tabellenform hätte dann nachfolgend dargestelltes Aussehen.

Regeln dieser Form werden beim Inferenzvorgang völlig analog ausgewertet, wobei die Teilerfüllungsgrade der nicht definierten Prämissen zu eins gesetzt werden. Die Matrixform ermöglicht eine besonders schnelle Erstellung der Regelbasis, da nur noch die Konklusionsterme definiert werden müssen. Die Umschaltung zwischen den beiden Darstellungsarten erfolgt über das Untermenü REGELBASIS. Das nachfolgende Bild zeigt den Regelbasis-Editor - bereits nach

Eingabe aller Regeln - für beide Darstellungsformen. In der Matrixform werden aus Platzgründen nicht die Bezeichner der linguistischen Terme, sondern die bei der Definition festgelegten Kürzel herangezogen. Die Formulierung der markierten Regel erfolgt dabei zusätzlich im Klartext in der Statuszeile. Ferner enthält die Statuszeile die Gewichtung dieser Regel in Prozent. Die Gewichtung läßt sich jedoch innerhalb der Matrixform nicht modifizieren.



Definition spezieller Regeln

Soll im Matrix-Modus eine Regel ausgelassen werden, so bleibt das entsprechende Matrixfeld frei. Die erweiterten Editieroptionen (Mehrfachauswahl von Einträgen) sind im Matrix-Modus nicht verfügbar.

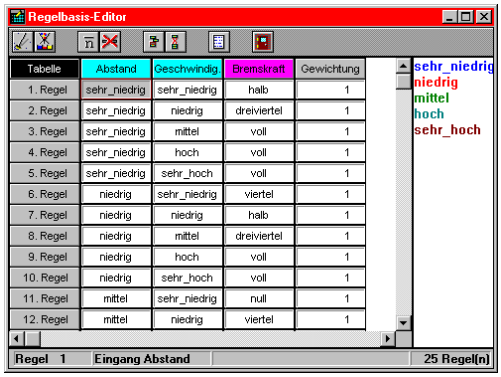
Wir können unserer Regelmatrix beispielsweise folgende Regeln entnehmen:

WENN Abstand = *sehr_niedrig* UND Geschwindigkeit = *sehr_niedrig*
DANN Bremskraft = *halb* (1. Zeile, 1. Spalte der Matrix)

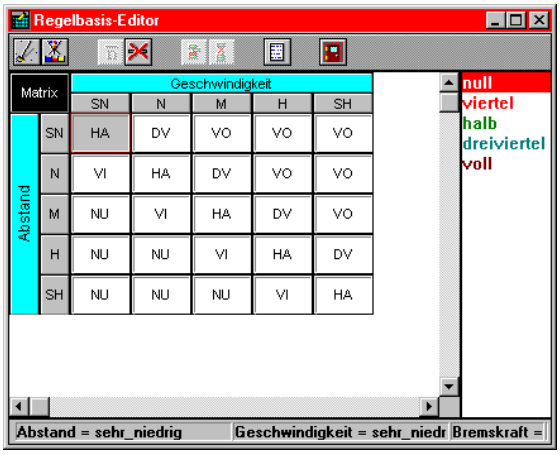
WENN Abstand = *niedrig* UND Geschwindigkeit = *niedrig*
DANN Bremskraft = *halb* (2. Zeile, 2. Spalte der Matrix)

WENN Abstand = *mittel* UND Geschwindigkeit = *hoch*
DANN Bremskraft = *dreiviertel* (3. Zeile, 4. Spalte der Matrix)

WENN Abstand = *sehr_hoch* UND Geschwindigkeit = *hoch*
DANN Bremskraft = *viertel* (5. Zeile, 4. Spalte der Matrix)



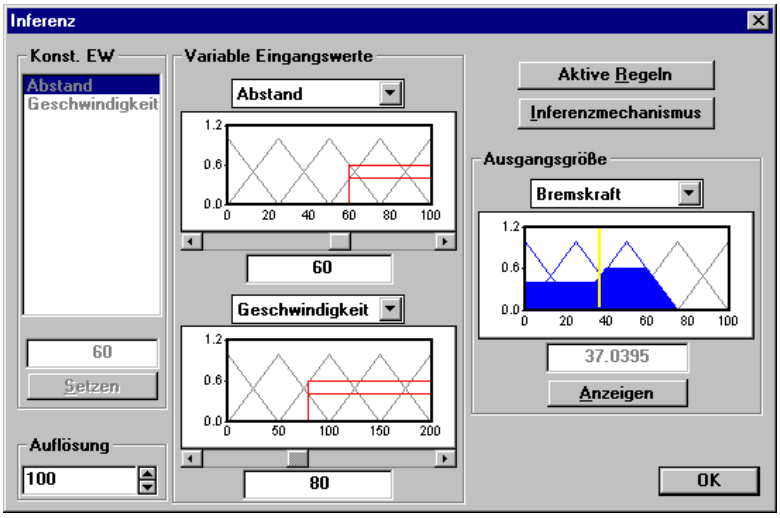
Regelbasis-Editor bei zwei Eingangsgrößen und einer Ausgangsgröße im Tabellen-Modus



Regelbasis-Editor im Matrix-Modus

Der komplette Datensatz für dieses Beispiel befindet sich unter dem Namen DEMO4.FUZ im Beispiel-Verzeichnis.

Zur Auswertung unserer Regelbasis begeben wir uns in den Inferenzdialog. Da wir in diesem Beispiel zwei Eingangsgrößen für das Regelwerk definiert haben, steht nunmehr auch das untere Anzeigefenster für die zweite Eingangsgröße zur Verfügung.



Inferenzdialog bei zwei Eingangsgrößen

FC-Kennfeld

Wechseln wir in die Kennfelddarstellung, so erhalten wir den bereits bekannten Dialog, der nunmehr allerdings das Kennfeld des Fuzzy-Systems enthält. Das Kennfeld ist mehrfarbig dargestellt, wobei niedrige Werte der Ausgangsgröße durch gelbe Farbgebung, höhere Werte durch einen zunehmenden Rotanteil gekennzeichnet werden.

Der Dialog bietet folgende Optionen:

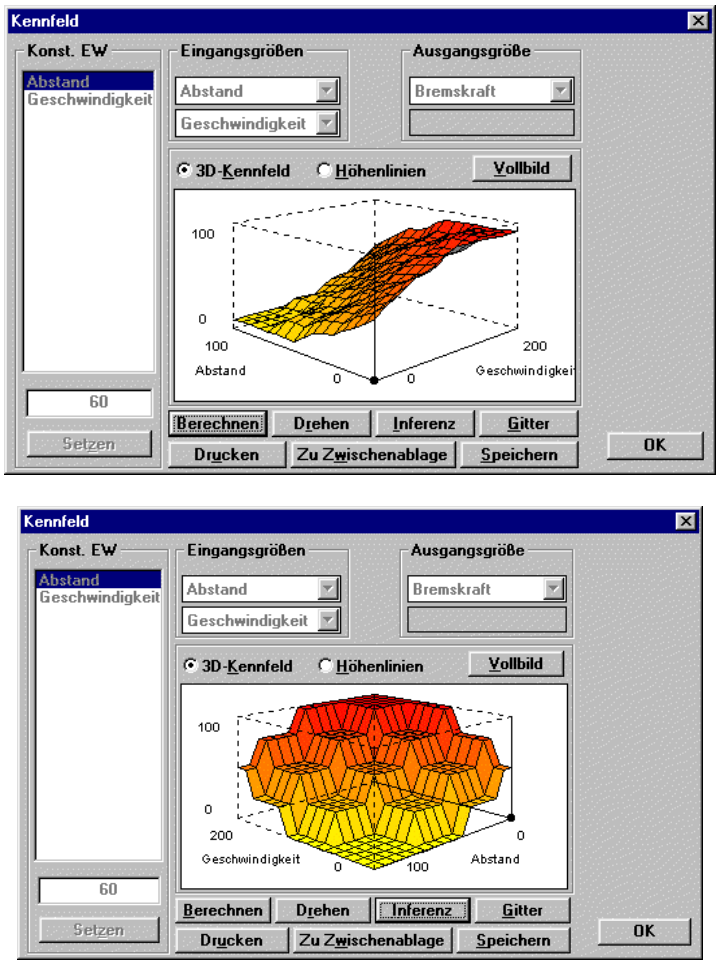
- Über *Drehen* kann das Kennfeld in Schritten von 45° gedreht werden, um die Beobachtungsrichtung möglichst optimal an die Orientierung des Kennfelds anzupassen.

  Da die eigentliche Kennfeldberechnung bei einer Drehung nicht neu erfolgen muß, erfolgt die Drehung in der Regel recht schnell.

- Die Schaltfläche *Speichern* ermöglicht ein Abspeichern der Ausgangsgrößenwerte in einer Datei vom Typ FWM. Diese Datei kann dann beispielsweise - nach Löschen der Projekt-Information und der ersten Zeile mit einem beliebigen Texteditor - unmittelbar mit Programmen wie etwa Mathematica zu 3D-Plots in Postscript-Qualität weiterverarbeitet werden.

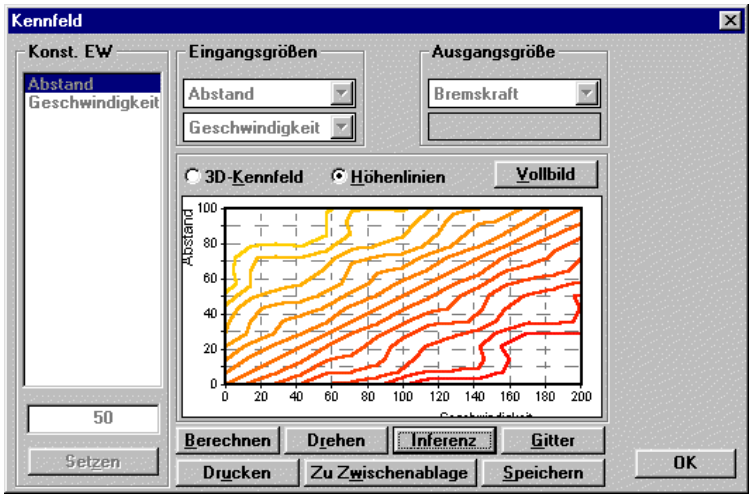
- Über *Gitter* wird die Diskretisierung des Kennfeldes, d. h. die Maschenanzahl des zugrundeliegenden Gitters und damit auch die erforderliche Rechenzeit, beeinflußt. Voreingestellt ist eine Stützpunktzahl von 15, erlaubt sind Werte zwischen 10 und 80.

- Der Auswahlpunkt *Höhenlinien* stellt das Kennfeld durch Höhenlinien dar. Wird diese zum *Vollbild* vergrößert, so werden die Mauskoordinaten zur Berechnung des Ausgangswertes herangezogen. Um möglichst aussagekräftige Höhenlinien zu erhalten, sollte die Anzahl der *Netzstücke pro Richtung* möglichst groß sein.



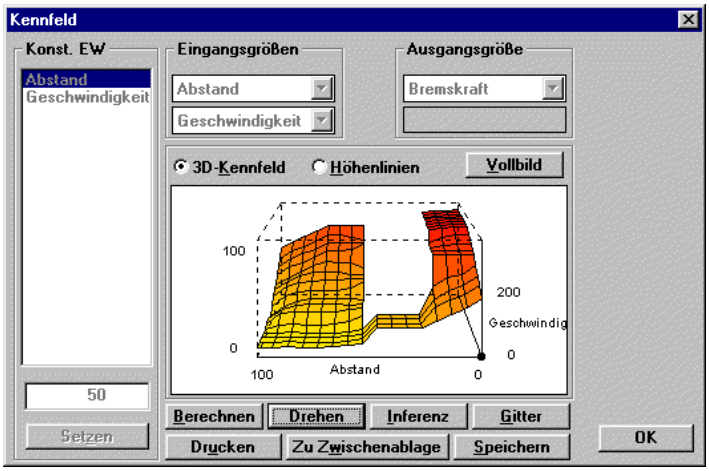
Kennfeld im Originalzustand (oben) und nach Drehung und Wechsel der Defuzzifizierungsmethode (unten)

Alle anderen Optionen verhalten sich genauso wie im Falle nur einer Eingangsgröße.



Höhenliniendarstellung

Es kann vorkommen, daß der berechnete Ausgangswert für die vorgegebenen Eingangswerte *uneindeutig* ist. Diese Uneindeutigkeit kommt immer dann zum Tragen, wenn keine Regel aktiv ist und für das Verhalten bei keiner aktiven Regel im Dialog Inferenzmechanismus und Defuzzifizierung *Alten Wert beibehalten* eingestellt worden ist. Derartige Bereiche des Kennfeldes werden ausgespart. Ein Beispiel dafür liefert die nachfolgende Grafik.



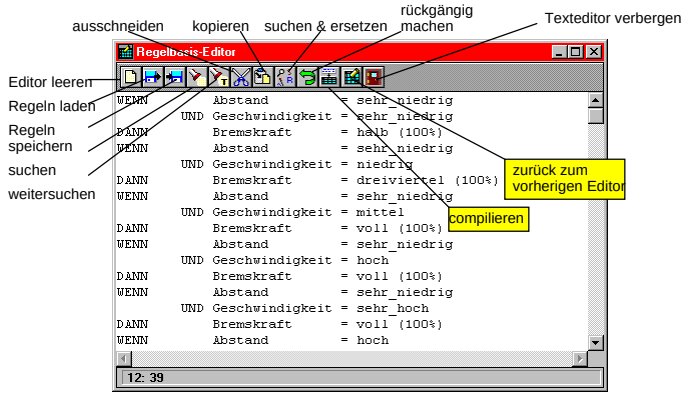
Kennfeld mit uneindeutigen Stellen

Regeln in Textform

FLOP kann, wie zuvor bereits erwähnt, Regeln in Textform bearbeiten. Um in den Texteditor zu gelangen, betätigen Sie die Schaltfläche *Texteditor* des Regelbasisfensters (Tabellen- bzw. Matrixeditor).

Sind im Tabellen- oder Matrixeditor keine Regeln definiert worden, so bleibt auch der Texteditor leer. Die Syntax der Regeln wird durch das nachfolgende Syntaxdiagramm erläutert.

Die Namen von linguistischen Variablen und deren Sets können durch einen Punkt an beliebiger Stelle abgekürzt werden. Es muß nur darauf geachtet werden, daß die Zeichen vor dem Punkt eine eindeutige Referenz zur Variable bzw. zum Term bilden. Hätte man die beiden Eingänge *Eingang1* und *Eingang2* definiert, so müßten diese vollständig ausgeschrieben werden, da sie sich nur im letzten Buchstaben voneinander unterscheiden.



Texteditor für Regeln

Vergleichsoperatoren wie z. B. = müssen in Leerzeichen eingeschlossen sein. Zum Negieren von Teilprämissen sind wahlweise die Operatoren

= NICHT

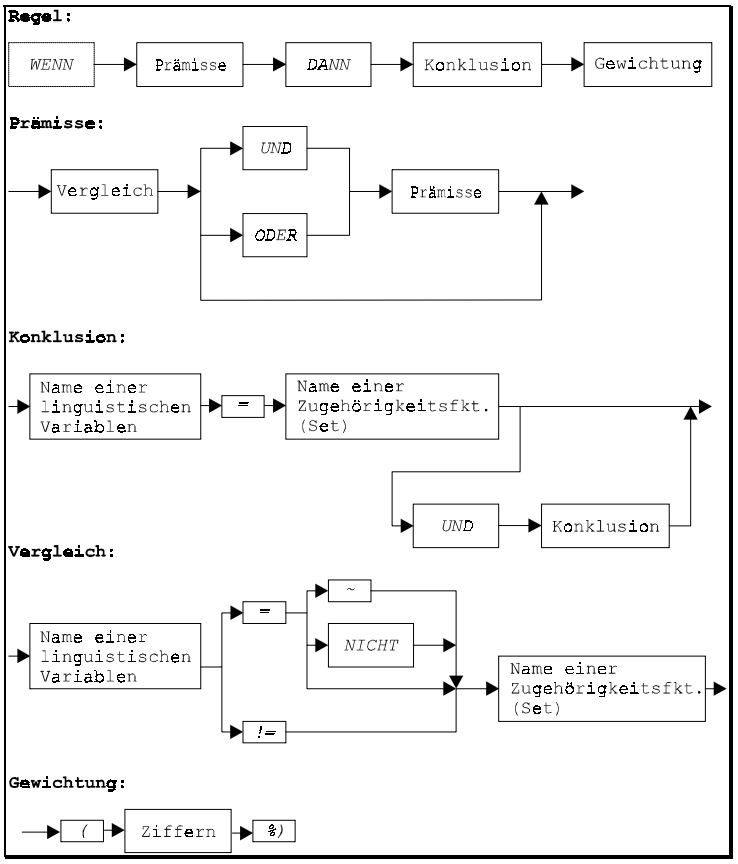
= ~

!=

zulässig. Bei syntaktischen Fehlern wird die entsprechende Stelle im Editor markiert und eine entsprechende Fehlermeldung ausgegeben. Folgende Merkmale zeichnen den Texteditor aus:

- Speichern und Laden von Textdateien (ASCII-Format, Defaultdateiendung ist RB) als Regeln. Beim Laden wird der geladene Text an der Stelle eingefügt, an der sich das Caret befindet.
- Alle Standard-Editor-Funktionen wie Suchen, Weitersuchen, Suchen und Ersetzen etc.

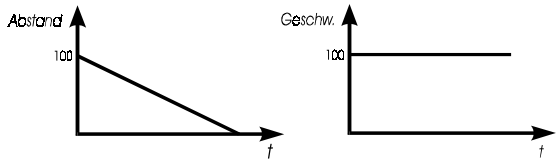
Wichtig: Bevor die modifizierte Regelbasis aktiv wird, muß sie compiliert werden! Dieses kann entweder über die *Compilieren*-Schaltfläche oder durch Rückkehr in einen anderen Regelbasis-Modus geschehen.



Syntaxdiagramm des Texteditors für Regeln

Simulation des Systems

Abschließend wollen wir das Verhalten unseres Systems für den Fall simulieren, daß der Abstand linear von 100 auf 0 abfällt, während die Geschwindigkeit einen konstanten Wert von 100 aufweist. Nachfolgendes Bild zeigt die zugehörigen Eingangsgrößenverläufe.



Eingangsgrößenverläufe für Beispielsimulation

Der zugehörige Datensatz befindet sich unter DEMO4.FSI im Beispiel-Verzeichnis und ist in nachfolgendem Listing auszugsweise dargestellt. Die Simulation liefert das nachfolgend dargestellte Ergebnis.

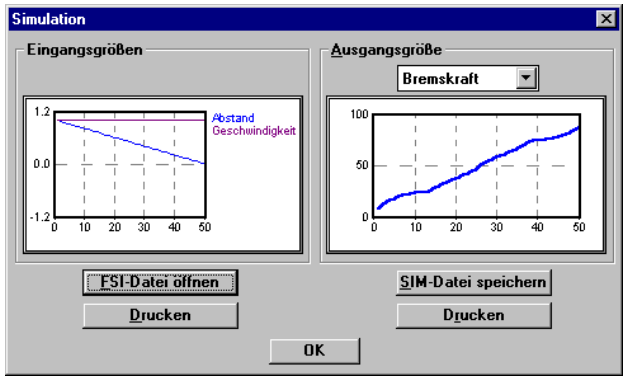
```
100  100
98   100
96   100
.    .
.    .
6    100
4    100
2    100
```

Simulationsdatensatz DEMO4.FSI (Ausschnitt)

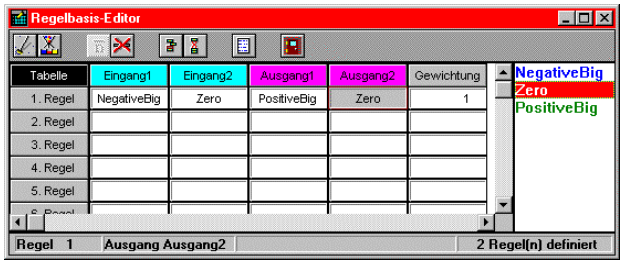
Die Vorgehensweise bei mehr als einer Ausgangsgröße ist völlig analog. In diesem Fall ist bei Durchführung von Inferenzschritten die jeweils anzuzeigende Ausgangsgröße über das entsprechende Listenfenster auszuwählen. Nehmen wir an, wir hätten die Eingangsgrößen *Eingang1* und *Eingang2* und die Ausgangsgrößen *Ausgang1* und *Ausgang2* gewählt, dann würde beispielsweise die Regel der Form

WENN *Eingang1* = *NegativeBig* UND *Eingang2* = *Zero*
DANN *Ausgang1* = *PositiveBig* UND *Ausgang2* = *Zero*

mit Hilfe des Regelbasis-Editors gemäß nachfolgendem Bild definiert. Da sich derartige Regeln jeweils in mehrere Regeln mit nur einer Ausgangsgröße aufsplitten lassen, werden sie programmintern auch mehrfach gezählt. Die Statuszeile im Regelbasiseditor gibt daher nach Festlegung obiger Regel die Meldung "2 Regel(n) definiert" aus.



Ergebnis der Simulation für Beispieldatensatz



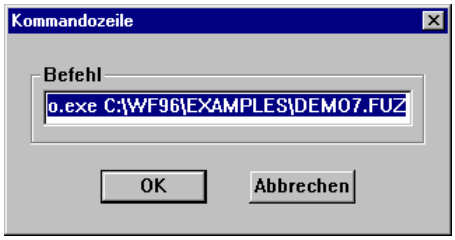
Definition von Regeln mit mehreren Ausgangsgrößen

Sonstige Optionen

Generierung von C-Quellcode

Sofern der C-Quellcodegenerator FALCO verfügbar ist, kann das aktuell bearbeitete Fuzzy-System direkt aus FLOP heraus in ANSI-C-Code überführt werden. Dazu wird über die Menüfolge DATEI | C-CODE GENERIEREN die Kom-

mandozeilenversion des Codegenerators aufgerufen. Hinweise dazu entnehmen Sie Kapitel 8 *Der Fuzzy-C-Code-Generator FALCO*.



Erzeugung von C-Code direkt aus der Fuzzy-Shell

Programmkonstanten

Maximale Anzahl linguistischer Variablen:	20
Maximale Anzahl linguistischer Terme (Fuzzy Sets) pro Variable:	10
Maximale Anzahl an Regeln:	350

Aufbau von FUZ-Dateien

Für die Arbeit mit FLOP ist eine Kenntnis der internen Dateistruktur nicht erforderlich. Sie kann jedoch notwendig sein, wenn beispielsweise anwendereigene Software auf diese Dateien zugreifen soll und wird daher in diesem Abschnitt im Detail erläutert.

Das nachfolgende Listing zeigt die Datei DEMO4.FUZ. Die mit einem Stern beginnenden Zeilen wurden zur besseren Gliederung der einzelnen Komponenten eingefügt und befinden sich *nicht* in der Originaldatei.

```
*********************************************
*                                           *
*             Datei-Information             *
*                                           *
*********************************************
!       Bremsvorgang auf der Autobahn
!       Eingangsgrößen: Abstand
!                       Geschwindigkeit
!       Ausgangsgröße: Bremskraft
```



```
!
*********************************************
*                                           *
*         Anzahl ling. Variablen            *
*                                           *
*********************************************
3
*********************************************
*                                           *
*          Eingangsgröße Abstand            *
*                                           *
*********************************************
Abstand
E
 0.00000000000000E+0000  1.00000000000000E+0002 1 1
5
sehr_niedrig
SN
1
0     25      1      0      0
niedrig
N
1
0     50      1     25      0
mittel
M
1
25    75      1     50      0
hoch
H
1
50    100      1      75      0
sehr_hoch
SH
1
75     100     1     100      0
*********************************************
*                                           *
*        Eingangsgröße Geschwindigkeit      *
*                                           *
*********************************************
Geschwindigkeit
E
 0.00000000000000E+0000  2.00000000000000E+0002 1 1
5
sehr_niedrig
SN
1
0     50     1     0     0
niedrig
```



```
N
1
0     100     1     50     0
mittel
M
1
50     150     1     100     0
hoch
H
1
100     200     1     150     0
sehr_hoch
SH
1
150     200     1     200     0
*********************************************
*                                           *
*          Ausgangsgröße Bremskraft         *
*                                           *
*********************************************
Bremskraft
A
 0.00000000000000E+0000  1.00000000000000E+0002 1 1
5
null
NU
1
0     25     1     0     0
viertel
VI
1
0     50     1     25     0
halb
HA
1
25     75     1     50     0
dreiviertel
DV
1
50     100     1     75     0
voll
VO
1
75     100     1     100     0
*********************************************
*                                           *
*                 Regelbasis                *
*                                           *
*********************************************
25
```



```
 1  1  3 1.00
 1  2  4 1.00
 1  3  5 1.00
 1  4  5 1.00
 2  3  4 1.00
 1  5  5 1.00
 2  1  2 1.00
 2  2  3 1.00
 2  4  5 1.00
 2  5  5 1.00
 3  1  1 1.00
 3  2  2 1.00
 3  3  3 1.00
 3  4  4 1.00
 3  5  5 1.00
 4  1  1 1.00
 4  2  1 1.00
 4  3  2 1.00
 4  4  3 1.00
 4  5  4 1.00
 5  1  1 1.00
 5  2  1 1.00
 5  3  1 1.00
 5  4  2 1.00
 5  5  3 1.00
**********************************************
*                                            *
*            Inferenzmechanismus             *
*                                            *
**********************************************
MAX_MIN
**********************************************
*                                            *
*              Defuzzifizierung              *
*                                            *
**********************************************
SP
**********************************************
*                                            *
*    Verhalten bei keiner aktiven Regel      *
*                                            *
**********************************************
HOLD
 0.00000000000000E+0000
```

Aufbau einer FUZ-Datei am Beispiel der Datei DEMO4.FUZ

Die Datei besteht aus folgenden Komponenten:

- Einer führenden Projekt-Information aus maximal acht Zeilen, die jeweils mit einem Ausrufezeichen beginnen.
- Einer Zeile mit der Gesamtzahl linguistischer Variablen (Eingangsgrößen + Ausgangsgrößen), hier 3.
- Den linguistischen Variablen, die wie folgt aufgebaut sind (hier erläutert anhand der Variablen *Abstand*):
  - Name der Variablen
  - Typ der Variablen (Ein- oder Ausgang), hier Eingang. "E" steht für Eingangsvariablen, "A" für Ausgangsvariablen.
  - Wertebereich der Variablen, hier 0 bis 100 mit anschließender Microcontrollerein- und -ausgangsbelegung. Diese beiden Ziffern sind für den Anwender uninteressant.
  - Anzahl der linguistischen Terme der Variablen, hier 5.
  - Linguistische Terme der Variablen, bestehend aus (hier erläutert anhand des Terms *sehr_niedrig*):
    - Name des Terms
    - Kürzel, hier *SN*
    - Typ der Zugehörigkeitsfunktion, hier Dreieck. Es bedeutet: 0 = Trapez, 1 = Dreieck, 2 = Singleton.
    - Charakteristische Punkte der Zugehörigkeitsfunktion in der Reihenfolge *Min, Max, Höhe, Knickpunkt 1, Knickpunkt 2*. Je nach Typ der Zugehörigkeitsfunktion werden nicht benötigte Werte zu null gesetzt.
- Der Regelbasis,die in der ersten Zeile die Anzahl der Regeln (hier 25) enthält. Danach folgen zeilenweise alle Regeln in Tabellenform, wobei die entsprechenden linguistischen Terme gemäß ihrer Reihenfolge bei der Definition durchnumeriert sind. Beispielsweise lauten die ersten drei Regeln

1 1 3:	WENN	*Abstand = sehr_niedrig*
	UND	*Geschwindigkeit = sehr_niedrig*
	DANN	*Bremskraft = halb*
1 2 4:	WENN	*Abstand = sehr_niedrig*
	UND	*Geschwindigkeit = niedrig*
	DANN	*Bremskraft = dreiviertel*

1 3 5:	WENN	*Abstand = sehr_niedrig*
	UND	*Geschwindigkeit = mittel*
	DANN	*Bremskraft = voll*

Enthält eine Regelzeile einen negativen Eintrag, so bedeutet dies, daß die entsprechende Teilprämisse negiert ist. Das letzte Element jeder Zeile stellt die Gewichtung der Regel dar.

- Dem Inferenzmechanismus. Es bedeuten

  MAX_MIN: Max-Min-Inferenz

  MAX_PROD: Max-Prod-Inferenz

- Der Defuzzifizierungsmethode. Es bedeuten

  SP: Originale Schwerpunktmethode

  SPM: Modifizierte Schwerpunktmethode

  SPN: Näherungsformel für Schwerpunktmethode (Schwerpunktmethode für Singletons)

  MHL: Maximum-Methode (links)

  MHR: Maximum-Methode (rechts)

  FOM: First of Maxima-Methode

  LOM: Last of Maxima-Methode

- Dem Systemverhalten bei keiner aktiven Regel. Es bedeuten

  HOLD: Der letzte gültige Ausgangswert wird beibehalten.

  DEFAULT: Es wird ein fester Vorgabewert ausgegeben.

Der Vorgabewert befindet sich in der nachfolgenden Zeile der Datei, und zwar auch dann, wenn wie hier der Mode HOLD gewählt wurde!

Für Dokumentationszwecke kann über die Menüfolge DATEI | DOKUMENT-DATEI GENERIEREN aus der undokumentierten FUZ-Datei eine Protokolldatei erzeugt werden. Diese trägt denselben Namen wie die FUZ-Datei, aber die Extension DOC. Diese Protokolldatei enthält ASCII-Text und kann daher mit jedem Textverarbeitungssystem weiterverarbeitet werden. Das nachfolgende Listing zeigt die aus der Datei DEMO4.FUZ erzeugte Dokumentdatei.

```
   WinFACT - Windows Fuzzy And Control Tools
      (C) Ingenieurbuero Dr. Kahlert 1998

Dokumentdatei zu: C:\WINFACT\DATEI\DEMO4.FUZ
erzeugt am  :  5.11.1997 um: 09:28
```

```
        3 linguistische Variablen

        2 Eingangs-Variable(n):
          - Abstand
          - Geschwindigkeit


        1 Ausgangs-Variable(n):
          - Bremskraft


        25 Regeln
        Max-Min-Inferenzmechanismus
        Methode der Defuzzyfizierung:
           Schwerpunkt

Eingangsgröße: Abstand  [  0.00000...100.00000]
-----------------------------------------------------------------
Name:        Form:    un. Grenze:  1.Knick:  2.Knick: ob. Grenze:
-----------------------------------------------------------------
sehr_niedrig Dreieck    0.00000     0.00000             25.00000
niedrig      Dreieck    0.00000    25.00000             50.00000
mittel       Dreieck   25.00000    50.00000             75.00000
hoch         Dreieck   50.00000    75.00000            100.00000
sehr_hoch    Dreieck   75.00000   100.00000            100.00000


Eingangsgröße: Geschwindigkeit  [  0.00000...200.00000]
-----------------------------------------------------------------
Name:        Form:    un. Grenze:  1.Knick:  2.Knick: ob. Grenze:
-----------------------------------------------------------------
sehr_niedrig Dreieck     0.00000    0.00000             50.00000
niedrig      Dreieck     0.00000   50.00000            100.00000
mittel       Dreieck    50.00000  100.00000            150.00000
hoch         Dreieck   100.00000  150.00000            200.00000
sehr_hoch    Dreieck   150.00000  200.00000            200.00000


Ausgangsgröße: Bremskraft  [  0.00000...100.00000]
-----------------------------------------------------------------
Name:        Form:    un. Grenze:  1.Knick:  2.Knick: ob. Grenze:
-----------------------------------------------------------------
null         Dreieck    0.00000     0.00000             25.00000
viertel      Dreieck    0.00000    25.00000             50.00000
halb         Dreieck   25.00000    50.00000             75.00000
dreiviertel  Dreieck   50.00000    75.00000            100.00000
voll         Dreieck   75.00000   100.00000            100.00000


Regelbasis:
-------------------------------------------------------------
Nr.  Abstand      Geschwindigkeit  => Bremskraft  Gewichtung
-------------------------------------------------------------
1    sehr_niedrig sehr_niedrig     => halb           1.0000
2    sehr_niedrig niedrig          => dreiviertel    1.0000
3    sehr_niedrig mittel           => voll           1.0000
4    sehr_niedrig hoch             => voll           1.0000
5    sehr_niedrig sehr_hoch        => voll           1.0000
6    niedrig      sehr_niedrig     => viertel        1.0000
7    niedrig      niedrig          => halb           1.0000
8    niedrig      mittel           => dreiviertel    1.0000
9    niedrig      hoch             => voll           1.0000
10   niedrig      sehr_hoch        => voll           1.0000
11   mittel       sehr_niedrig     => null           1.0000
12   mittel       niedrig          => viertel        1.0000
```



```
13   mittel       mittel          => halb          1.0000
14   mittel       hoch            => dreiviertel   1.0000
15   mittel       sehr_hoch       => voll          1.0000
16   hoch         sehr_niedrig    => null          1.0000
17   hoch         niedrig         => null          1.0000
18   hoch         mittel          => viertel       1.0000
19   hoch         hoch            => halb          1.0000
20   hoch         sehr_hoch       => dreiviertel   1.0000
21   sehr_hoch    sehr_niedrig    => null          1.0000
22   sehr_hoch    niedrig         => null          1.0000
23   sehr_hoch    mittel          => null          1.0000
24   sehr_hoch    hoch            => viertel       1.0000
25   sehr_hoch    sehr_hoch       => halb          1.0000
```

Dokumentdatei zur Datei DEMO4.FUZ

8 Der Fuzzy-C-Code-Generator FALCO

Leistungsumfang

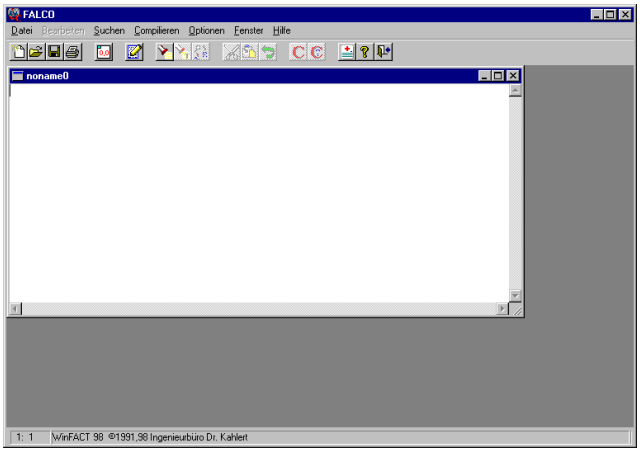
Der C-Quellcodegenerator FALCO (**F**uzzy **A**ppl**l**ication C-**Co**degenerator) ermöglicht die Erzeugung von ANSI-C-Code für Fuzzy-Systeme, die mit Hilfe der WinFACT-Fuzzy-Shell FLOP entwickelt wurden. Der C-Quellcode besteht im wesentlichen aus einer C-Funktion, die dann in anwendereigene Software eingebunden bzw. zur Programmierung einer Zielhardware benutzt werden kann. FALCO zeichnet sich durch folgende Leistungsmerkmale aus:

- Komfortabler Editor für FUZ- und C-Dateien mit MDI-Schnittstelle und allen herkömmlichen Editorfunktionen wie Editieren, Kopieren, Löschen, Suchen und Ersetzen usw.
- Die parallele Bearbeitung mehrerer Systeme möglich.
- Der generierte C-Code besteht aus lediglich einer Datei - es sind keine zusätzlichen Bibliotheken erforderlich. Der Code ist sehr kompakt und enthält nur die für das jeweils compilierte Fuzzy-System nötigen Anweisungen wie Inferenzmechanismus, Defuzzifizierungsmethode usw.
- Die Datentypen für die linguistischen Variablen (Ein- und Ausgangsgrößen des Fuzzy-Systems) und Zugehörigkeitswerte sind vom Anwender frei wählbar (8 Bit - Integer, 16 Bit - Integer, Float, ...).
- Automatische Erzeugung eines C-Rahmenprogramms mit Debugging-Möglichkeit.
- Compilierung und Ausführung des generierten C-Codes aus der Entwicklungsumgebung heraus über verschiedene Compiler (z. B. für PC/Zielhardware) möglich.
- Kommandozeilenaufruf (z. B. aus der WinFACT-Fuzzy-Shell FLOP heraus) möglich

Entwicklungsumgebung

Bildschirmaufbau

Das nachfolgende Bild zeigt das Anwendungsfenster von FALCO unmittelbar nach dem Aufruf des Programms.



Hauptfenster von FALCO nach dem Aufruf

Das Anwendungshauptfenster enthält neben den Standardkomponenten

- eine horizontale Toolbar unter dem Hauptmenü zum Direktzugriff auf die wichtigsten Funktionen,
- ein oder mehrere Dokumentfenster, die sowohl eingelesene FUZ-Dateien als auch daraus generierten C-Code enthalten können,
- eine Statuszeile am unteren Fensterrand, die die aktuelle Cursorposition innerhalb des aktiven Dokumentfensters anzeigt.

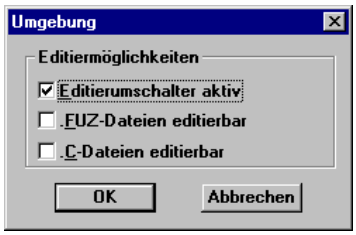
Editoroptionen

Laden von FUZ-Dateien

Das zu verarbeitende Fuzzy-System muß sich in einer Datei mit der Extension FUZ befinden, wie sie von der WinFACT-Fuzzy-Shell FLOP generiert wird. Die Datei kann über die Menüfolge DATEI | ÖFFNEN in das aktive Dokumentfenster geladen werden. Eine eventuell im Dokumentfenster vorhandene Datei wird dabei überschrieben. Soll ein neues Dokumentfenster für das Fuzzy-System generiert werden, so ist dieses zunächst über DATEI | NEU zu erzeugen. Ein Löschen des aktuellen Fensterinhalts ist über BEARBEITEN | KOMPLETT LÖSCHEN möglich. Ein Abspeichern eventuell modifizierter Dateien erlaubt die Menüfolge DATEI | SPEICHERN. Ferner kann die im aktiven Dokumentfenster befindliche Datei über DATEI | DRUCKEN auf dem Standarddrucker ausgegeben werden.

Bearbeiten von Dateien

In der Regel ist ein Editieren von FUZ-Dateien oder auch C-Dateien nicht erforderlich. Um jedoch dem erfahrenen Anwender die Möglichkeit zu geben, sowohl die Fuzzy-Systemdateien als auch den generierten C-Quellcode bei Bedarf anzupassen, enthält die FALCO-Entwicklungsumgebung einen kompletten Editor. Um versehentliche Änderungen von Dateien zu verhindern, kann der Editiermodus - für FUZ- und C-Dateien getrennt - gesperrt werden. Den entsprechenden Dialog erhält man über die Menüfolge OPTIONEN | UMGEBUNG.



Dialog zum Sperren und Freigeben des Editiermodus

Sofern die Option *Editierumschalter aktiv* aktiviert wurde, kann die Editiermöglichkeit unabhängig vom Dateityp des aktiven Fensters jederzeit über die Toolbar freigegeben bzw. gesperrt werden. Standardmäßig ist die Editiermöglichkeit für beide Dateitypen gesperrt. Eine veränderte Einstellung kann über die Menüfolgen OPTIONEN | SPEICHERN, OPTIONEN | SPEICHERN UNTER

bzw. OPTIONEN | LADEN jederzeit abgespeichert bzw. wieder geladen werden. Dabei werden alle Einstellungen der Dialoge *Umgebung und C-Code-Generierung* (s. Abschnitt *Voreinstellungen*) gesichert.

Ist der Editiermodus freigegeben, so stehen dem Anwender alle herkömmlichen Editierfunktionen zur Verfügung (siehe z. B. [1]). Dazu zählen insbesondere

Editier-funktionen

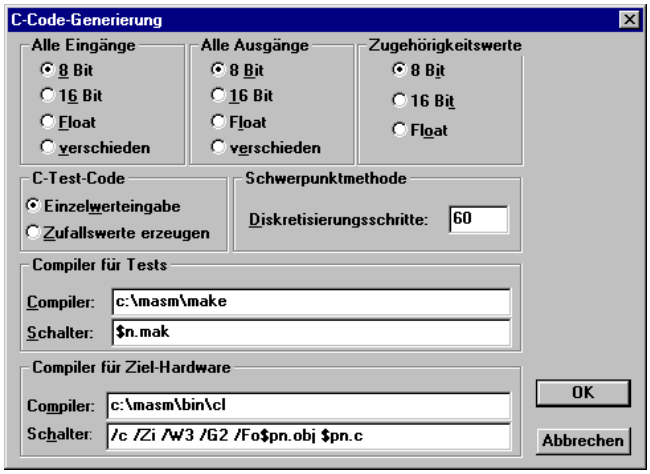
- das Ausschneiden von Textblöcken in die Zwischenablage,
- das Kopieren von Textblöcken in die Zwischenablage,
- das Einfügen von Textblöcken aus der Zwischenablage,
- das Suchen und eventuelle Ersetzen von Textblöcken,
- das Rückgängigmachen der letzten Änderung.

Alle Optionen befinden sich in den Untermenüs BEARBEITEN bzw. SUCHEN.

C-Quellcodegenerierung

Voreinstellungen

Vor der Generierung des C-Codes sind in der Regel einige Voreinstellungen zu treffen, die insbesondere die Auflösung, d. h. den verwendeten C-Datentyp für die linguistischen Variablen und die Zugehörigkeitswerte betreffen. Diese Einstellungen werden über OPTIONEN | CODE-GENERATOR vorgenommen:



Dialog zur Wahl der Datentypen für linguistische Variablen und Zugehörigkeitswerte

Es stehen als Datentypen zur Verfügung:

Datentypen

- 3..8 Bit - Integer (unsigned char)
- 9..16 Bit - Integer (unsigned short)
- Float

Voreingestellt ist für alle Größen eine Auflösung von 8 Bit. Sollen für einzelne Ein- bzw. Ausgangsgrößen unterschiedliche Datentypen gewählt werden, so ist das Feld *verschieden* zu aktivieren. In diesem Fall können nach dem Aufruf der eigentlichen Quellcodegenerierung die einzelnen Datentypen über einen Eingabedialog gewählt werden:

Wahl der Auflösung bei unterschiedlichen Datentypen für einzelne Ein- bzw. Ausgangsgrößen

Der Optionen-Dialog ermöglicht weiterhin die Vorgabe des Kommandozeilen-Aufrufs und eventueller Schalter für die benutzten C-Compiler. Dabei können zwei unterschiedliche Compiler - beispielsweise ein PC-Compiler und ein Compiler für die Zielhardware - angegeben werden. Die beiden vorgegebenen Aufrufe werden benutzt, wenn der generierte C-Code aus FALCO heraus compiliert werden soll.

Innerhalb der *Compiler*- bzw. *Schalter*-Editierfelder kann, wie in obigem Dialog geschehen - der Name der aktuellen Datei automatisch eingefügt werden. Dazu dienen die Kürzel

$n: fügt beim Aufruf den Dateinamen (ohne Extension) ein,

$pn: fügt beim Aufruf den Dateipfad und -namen (ohne Extension) ein.

Für obige Vorgaben und die Beispieldatei DEMO4.FUZ im Verzeichnis C:\WINFACT\DATEI würde dies bedeuten:

Bei Aufruf von COMPILIEREN | TEST-COMPILAT GENERIEREN (s. nachfolgenden Abschnitt) wird die Kommandozeile

c:\masm\make demo4.mak

ausgeführt, bei Aufruf von COMPILIEREN | ZIELHARDWARE-COMPILAT GENERIEREN entsprechend die Kommandozeile

c:\masm\bin\cl /c /Zi /W3 /G2 /Fo c:\winfact\datei\demo4.obj c:\winfact\datei\demo4.c.

Die Einstellungen im mittleren Gruppenfenster des Optionen-Dialogs (C-Code zum Testen) sind nur von Bedeutung, wenn neben der eigentlichen C-Funktion auch Testcode zum Debuggen erzeugt werden soll. In diesem Fall besteht wahlweise die Möglichkeit, scharfe Eingangswerte über die Tastatur einzulesen und diese dann auswerten zu lassen oder aber die scharfen Eingangswerte zufällig erzeugen zu lassen.

Schließlich kann im Falle der Defuzzifizierung nach der Schwerpunktmethode die Auflösung für die numerische Integration zur Bestimmung des Flächenschwerpunktes gewählt werden. Voreingestellt ist ein Wert von 60 Diskretisierungsschritten. Der hier eingestellte Wert beeinflußt in erheblichem Maße die Ausführungsgeschwindigkeit der Defuzzifizierung und damit des gesamten Codes und sollte daher mit Bedacht gewählt werden!

Tip: Im Hinblick auf einen möglichst schnellen Code sollte die Schwerpunktmethode durch die Höhenmethode (Schwerpunktmethode für Singletons) ersetzt werden, die bei nahezu gleichen Ergebnissen erheblich schneller arbeitet!

Codegenerierung

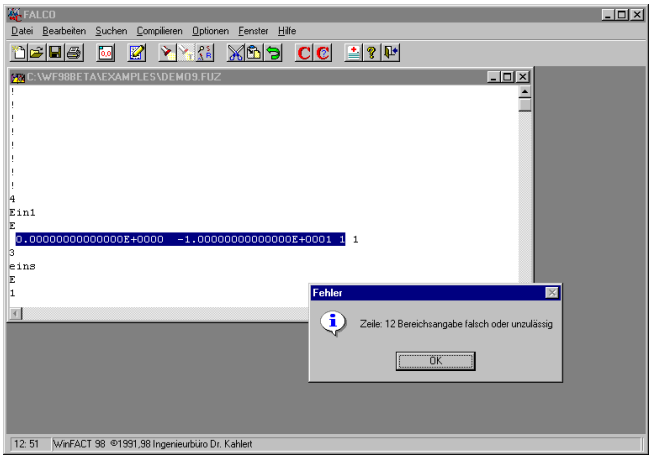
Starten der Codegenerierung

Nach Beendigung der Voreinstellungen kann die eigentliche Codegenerierung gestartet werden, die sich jeweils auf die FUZ-Datei im aktiven Dokumentfenster bezieht. Die Codegenerierung kann auf verschiedene Weisen erfolgen:

- Es kann nur die eigentliche C-Funktion erzeugt werden, die das Verhalten des Fuzzy-Systems beschreibt (Menüfolge COMPILIEREN | C-CODE GENERIEREN).
- Es kann zusätzlich ein "Rahmenprogramm" erzeugt werden, das die Auswertung der Funktion für vom Anwender vorgegebene bzw. zufällig erzeugte Eingangswerte ermöglicht und somit ein Austesten des Codes erlaubt (Menüfolge COMPILIEREN | C-TEST-CODE GENERIEREN).

Überprüfung auf Zulässigkeit

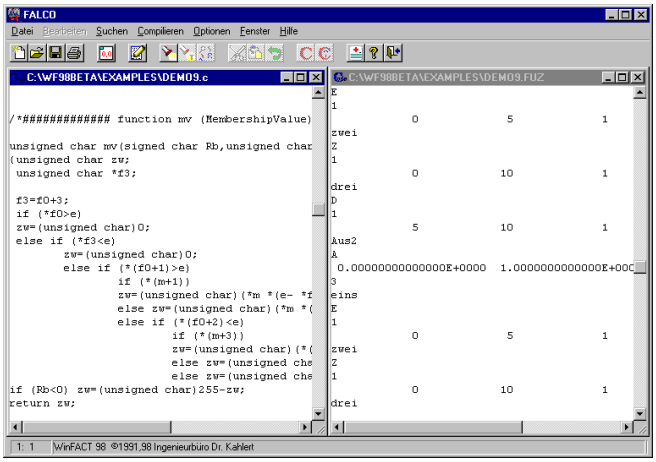
Nach dem Starten des Compiliervorgangs wird die Datei zunächst auf ihre Zulässigkeit geprüft. Diese Prüfung betrifft einerseits die Dateisyntax, andererseits aber auch Plausibilitätsprüfungen wie etwa die Konsistenz von Parameterkombinationen usw. Diese Prüfung ist insbesondere dann wertvoll, wenn eine FUZ-Datei manuell erstellt oder modifiziert wurde. Werden Fehler entdeckt, erfolgt eine entsprechende Fehlermeldung und der Cursor wird automatisch an die Fehlerstelle gesetzt.



Konsistenzprüfung der FUZ-Datei

Darstellung des generierten Codes

Der erzeugte C-Quellcode erscheint in einem separaten Dokumentfenster, das denselben Titel wie die zugrundeliegende FUZ-Datei (jedoch mit der Extension C) trägt.



Hauptfenster mit FUZ- (rechts) und generierter C-Datei (links)

Die C-Code-Datei wird nach der C-Codegenerierung automatisch gespeichert.

Codegenerierung aus der Kommandozeile

Wird FALCO von einer Kommandozeile, beispielsweise der des Windows-Dateimanagers oder aus der Fuzzy-Shell FLOP heraus aufgerufen, so kann ihm über Parameter mitgeteilt werden, aus welcher FUZ-Datei C-Code generiert werden soll:

Der allgemeine Aufruf lautet:

falco FUZ-Datei [C-Datei][/e#n1,#n2,..][/a#n1,n2,..][/mv#m][/d#k][/te od. /tz].

Die Parameter haben folgende Bedeutung:

Parameter	Bedeutung
FUZ-Datei	Name der FUZ-Datei. Diese Angabe ist zwingend!
C-Datei	Name der zu erzeugenden C-Datei. Falls diese Angabe fehlt, wird der Name der FUZ-Datei mit der Extension C gewählt.
/e#n1,#n2,....	Legt die Bitbreite der Eingänge in der Reihenfolge fest, in der die Variablen in der FUZ-Datei definiert wurden. Für die Beispieldatei DEMO4.FUZ erhält der Eingang *Geschwindigkeit* die Bitbreite #n1, die linguistische Variable *Abstand* wird dagegen mit #n2 Bit aufgelöst. Werden hinter /e weniger Werte angegeben, als Eingänge vorliegen, so werden die restlichen Eingänge auf den zuletzt angegebenen Wert gesetzt. Der Parameter /e8 setzt also alle Eingänge auf 8 Bit.
/a#n1#n2,.....	Legt die Bitbreite der Ausgänge fest. Es gilt das Gleiche wie bei den Eingängen!
/mv#m	Legt die Auflösung der Zugehörigkeitsfunktion fest.
/d#k	Gibt die Anzahl der Diskretisierungsschritte an (wird nur bei der Schwerpunktmethode und der modifizierten Schwerpunktmethode verwendet).
/te	Teilt FALCO mit, daß Einzelschritt-Test-Code erzeugt werden soll.
/tz	Teilt FALCO mit, daß Test-Code mit Zufallswerten erzeugt werden soll.

Zulässige Wertebereiche:

#n1,#n2,....	mögliche Werte: 3..16, 32
	3..8 Bit (unsigned char)
	9..16 Bit (unsigned short)
	32 Bit (float)
#m	mögliche Werte: 8, 16, 32
#k	Je größer, desto genauer aber langsamer ist die Schwerpunktbestimmung!

Struktur des erzeugten C-Quellcodes

Der erzeugte C-Code besteht im wesentlichen aus einer Funktion mit dem Funktionsnamen *fuzzy*. Eingangsparameter der Funktion sind die scharfen Eingangswerte des Fuzzy-Systems, Ausgangsparameter die scharfen Ausgangswerte. Der Funktionskopf hat für ein System mit *m* Eingängen und *n* Ausgängen die folgende Struktur:

Funktions-deklaration

```
void fuzzy(char *changed,
           Inputtyp1 Input1, ...,
           Inputtypm Inputm,
           Outputtyp1 *Output1, ...,
           Outputtypm *Outputm)
```

Die Input- bzw. Outputtypen hängen ab von den gewählten Datentypen für die einzelnen linguistischen Variablen. Die Bezeichner der Ein- und Ausgangsgrößen werden entsprechend den Bezeichnern innerhalb der FUZ-Datei gewählt, erhalten aber zusätzlich eine Kennummer sowie die Endung *Value*.

Die Variable *changed* wird in der Regel nicht benötigt[1] ; sie gibt für jede Ausgangsgröße getrennt an, ob beim aktuellen Aufruf der Funktion mit den entsprechenden Eingangswerten für diese Ausgangsgröße mindestens eine Regel aktiv war und somit eine Neuberechnung dieses Ausgangs erfolgt ist. In diesem Fall wird das entsprechende Feld von *changed* auf 1 gesetzt. War keine Regel aktiv, so wird je nach Einstellung innerhalb der FUZ-Datei der alte Ausgangswert dieser Größe beibehalten bzw. ein Defaultwert angenommen. Das entsprechende Feld von *changed* behält in diesem Fall seinen Wert bei, den es beim Aufruf der Funktion hatte (der Anwender ist ggf. also selbst dafür verantwortlich, daß das Feld vor dem Aufruf der Funktion auf 0 gesetzt wird!).

[1] Die Variable wird WinFACT-intern für den BORIS-Autocode-Generator benutzt.

Beispiel:

Für ein Fuzzy-System mit den beiden Float-Eingangsgrößen *Druck* und *Temperatur* und der Float-Ausgangsgröße *Ventil* wird die folgende Funktionsdeklaration erzeugt:

```
void fuzzy(char *changed,
           float Druck1Value,
           float Temperatur2Value,
           float *Ventil3Value)
```

Diese Funktion kann aus einem Anwenderprogramm beispielsweise wie folgt aufgerufen werden:

Aufruf der Funktion

```
float Input1, Input2, Output;
char Dummy[1];
.
.
Input1 = 25.5;
Input2 = 35.2;
fuzzy(Dummy, Input1, Input2, &Output);
```

Nachfolgendes Listing zeigt die Struktur des erzeugten C-Codes für ein Beispiel mit zwei Eingängen, einem Ausgang, 25 Regeln, max-min-Inferenz und Defuzzifizierung nach der Schwerpunktmethode (erzeugt aus der Beispieldatei DEMO4.FUZ). Als Datentyp für Ein- und Ausgangsgrößen wurde hier 8 Bit - Integer gewählt.

```
/* WinFACT - Windows Fuzzy And Control Tools
// (C) Ingenieurbuero Dr. Kahlert 1991,96
//
// File C:\WINFACT\WF30\EXAMPLES\DEMO4.c generated at :  9. 4.1996   16:44
//
// 3  linguistic variables
//
// 2  Input(s):
//   - Abstand1value (unsigned char)
//   - Geschwindigkei2value (unsigned char)
//
// 1  Output(s):
//   - Bremskraft3value (unsigned char)
//
// 25 rules
// Max-Min-inference
// Method of defuzzyfication:
// center of gravity  built up through 60 steps
*/

#define Maxreg 25
```

```
#define Anzvar 3
#define Anzein 2
#define Anzaus 1
#define Maxsets 6
#define Diskret 60


/*############## Declaration of the fuzzy sets and rules ###########*/

static unsigned char Abstand1[5][4]={
{( char)         0,( char)         0,( char)         0,( char)        64},
{( char)         0,( char)        64,( char)        64,( char)       128},
{( char)        64,( char)       128,( char)       128,( char)       191},
{( char)       128,( char)       191,( char)       191,( char)       255},
{( char)       191,( char)       255,( char)       255,( char)       255}};

static unsigned char Geschwindigkei2[5][4]={
{( char)         0,( char)         0,( char)         0,( char)        64},
{( char)         0,( char)        64,( char)        64,( char)       128},
{( char)        64,( char)       128,( char)       128,( char)       191},
{( char)       128,( char)       191,( char)       191,( char)       255},
{( char)       191,( char)       255,( char)       255,( char)       255}};

static unsigned char Bremskraft3[5][4]={
{( char)         0,( char)         0,( char)         0,( char)        64},
{( char)         0,( char)        64,( char)        64,( char)       128},
{( char)        64,( char)       128,( char)       128,( char)       191},
{( char)       128,( char)       191,( char)       191,( char)       255},
{( char)       191,( char)       255,( char)       255,( char)       255}};

static char Abstand1slope[5][4]={
{(char)255,(char)0,(char)4,(char)0},
{(char)4,(char)0,(char)4,(char)0},
{(char)4,(char)0,(char)4,(char)0},
{(char)4,(char)0,(char)4,(char)0},
{(char)4,(char)0,(char)255,(char)0}};

static char Geschwindigkei2slope[5][4]={
{(char)255,(char)0,(char)4,(char)0},
{(char)4,(char)0,(char)4,(char)0},
{(char)4,(char)0,(char)4,(char)0},
{(char)4,(char)0,(char)4,(char)0},
{(char)4,(char)0,(char)255,(char)0}};

static char Bremskraft3slope[5][4]={
{(char)255,(char)0,(char)4,(char)0},
{(char)4,(char)0,(char)4,(char)0},
{(char)4,(char)0,(char)4,(char)0},
{(char)4,(char)0,(char)4,(char)0},
{(char)4,(char)0,(char)255,(char)0}};

signed char RB[25][3]={
{1,1,3},
{1,2,4},
{1,3,5},
{1,4,5},
{1,5,5},
{2,1,2},
{2,2,3},
{2,3,4},
{2,4,5},
{2,5,5},
{3,1,1},
{3,2,2},
{3,3,3},
{3,4,4},
```



```
{3,5,5},
{4,1,1},
{4,2,1},
{4,3,2},
{4,4,3},
{4,5,4},
{5,1,1},
{5,2,1},
{5,3,1},
{5,4,2},
{5,5,3}};


unsigned char GW[25][1]={
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10},
{10}};


/*############# function mv (MembershipValue) ####################*/

unsigned char mv(signed char Rb,unsigned char *f0,char *m,unsigned char e)
{unsigned char zw;
 unsigned char *f3;

 f3=f0+3;
 if (*f0>e)
 zw=(unsigned char)0;
 else if (*f3<e)
  zw=(unsigned char)0;
  else if (*(f0+1)>e)
    if (*(m+1))
    zw=(unsigned char)(*m *(e- *f0)+(e- *f0)/ *(m+1));
    else zw=(unsigned char)(*m *(e- *f0));
    else if (*(f0+2)<e)
      if (*(m+3))
      zw=(unsigned char)(*(m+2) *(*f3-e)+(*f3-e)/ *(m+3));
      else zw=(unsigned char)(*(m+2) *(*f3-e));
      else zw=(unsigned char)255;
if (Rb<0) zw=(unsigned char)255-zw;
return zw;
}
```

```
/*###################### function fuzzy #########################*/

void fuzzy(char *changed,
  unsigned char Abstand1value,
  unsigned char Geschwindigkei2value,
  unsigned char *Bremskraft3value)
{
 unsigned char zw[Anzein];
 unsigned char erg[Anzaus][Maxsets+2];
 int   i,j;
 unsigned char rb;
 unsigned char y,ug,og,delta;
 unsigned long z,n;

 unsigned char hl,mue;


/*############ Initialisation of the local variable erg[][] ########*/

  for(j=0;j<Anzaus;j++)
      {erg[j][0]=Maxsets;
       for(i=1;i<Maxsets+2;erg[j][i++]=0);
     };

/*######### Get the active rules with highest membershipvalue
############*/

  for(i=0;i<Maxreg;i++)
  { rb=(RB[i][0]>0)? RB[i][0]-1: ~RB[i][0];
  zw[0]=(RB[i][0]==0)? (unsigned char)255 :mv(
    RB[i][0],
    (unsigned char*)Abstand1[rb],
    Abstand1slope[rb],
    (unsigned char) Abstand1value);
  if (zw[0]>0)
    { rb=(RB[i][1]>0)? RB[i][1]-1: ~RB[i][1];
    zw[1]=(RB[i][1]==0)? (unsigned char)255 :mv(
      RB[i][1],
      (unsigned char*)Geschwindigkei2[rb],
      Geschwindigkei2slope[rb],
      (unsigned char) Geschwindigkei2value);
    if (zw[1]>0)
    {
    for(j=1;j<Anzein;j++)
      zw[0]=(zw[j]>zw[0])? zw[0] : zw[j];
    zw[0]=zw[0]/10*GW[i][0];
    for(j=0;j<Anzaus;j++)
      if (RB[i][2+j]!=0)
        { rb=(RB[i][2+j]>0)? RB[i][2+j]-1: ~RB[i][2+j];
          if(erg[j][rb+2]<zw[0]) erg[j][rb+2]=zw[0];
          if(erg[j][0]>(unsigned char)rb+2)erg[j][0]=(unsigned char)(rb+2);
          if(erg[j][1]<(unsigned char)rb+2)erg[j][1]=(unsigned char)(rb+2);}
    }
    }};

/*################### Part of the defuzzyfication ######################*/

if (erg[0][1]) {changed[0]=1;
z=n=(long int)0;
ug=(unsigned char)Bremskraft3[erg[0][0]-2][0];
og=(unsigned char)Bremskraft3[erg[0][1]-2][3];
delta=(unsigned char)((og-ug)/(Diskret-1));
if (delta==(unsigned char)0) delta=(unsigned char)1;
for(y=ug;(y<og)&&(y<(unsigned char)(y+delta));)
  {y+=delta;
  mue=(unsigned char)0;
```



```
   for(i=erg[0][0];i<=erg[0][1];i++)
    {hl=mv(1,
      (unsigned char*)Bremskraft3[i-2],
      Bremskraft3slope[i-2],
      (unsigned char)y);
     hl=hl>erg[0][i] ? erg[0][i]:hl;
    mue=hl>mue?hl:mue;};
  z+=(long int)(y)*(mue);
  n+=(long int)((mue));
  }
*Bremskraft3value=(unsigned
char)((n==0)?(erg[0][erg[0][0]]+erg[0][erg[0][1]]):z/n);
}
}
```

Aus der Datei DEMO4.FUZ erzeugter C-Code

Der Code weist die folgende Struktur auf:

- Programmkopf mit einer Übersicht über das verarbeitete Fuzzy-System
- Konstantendeklarationen (Anzahl der Ein- und Ausgänge, Anzahl der Regeln usw.)
- Definition der Fuzzy-Sets

  Hier werden neben den charakteristischen Punkten (linker und rechter Randpunkt, Knickpunkte) auch die Flankensteigungen abgelegt, da diese zur schnellen Berechnung von Zugehörigkeitswerten benötigt werden. Wurde für eine linguistische Variable ein Integer-Datentyp definiert, so werden die Originalwerte aus der FUZ-Datei auf den entsprechenden Wertebereich (z. B. 0..255 bei 8 Bit-Auflösung) umnormiert.
- Definition der Regelbasis und der Gewichtungen der Regeln
- Funktion *mv* zur Berechnung von Zugehörigkeitswerten
- Funktion *fuzzy* zur Fuzzifizierung, Inferenz und Defuzzifizierung. Die interne Realisierung dieser Schritte hängt ab von den gewählten Datentypen für die Ein- und Ausgänge, der Inferenzart und der Defuzzifizierungsmethode.
- Falls die Option *C-Testcode generieren* gewählt wurde, schließt sich ein einfaches Hauptprogramm an, welches das Testen des Codes anhand scharfer Eingangswerte ermöglicht. Außerdem sind mit Hilfe dieses Testcodes Zeitmessungen möglich, sofern die C-Bibliothek time.h vorhanden ist.

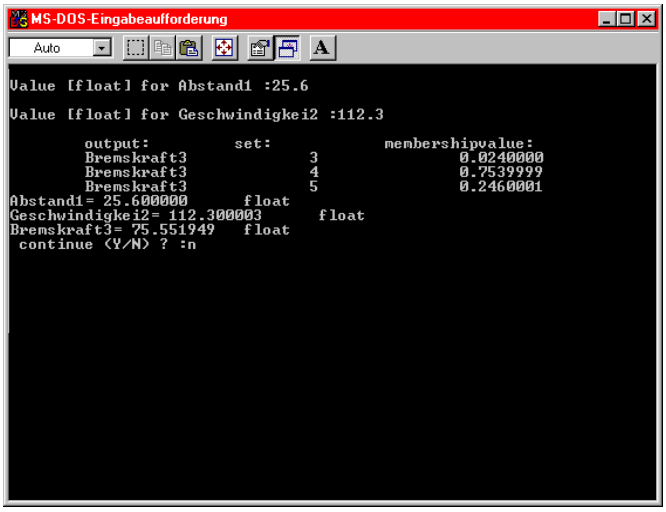
```
/*###################### main - function ##########################*/

int main(void)
```

```
{
 char changed[1];
 unsigned char Abstand1value;
 unsigned char Geschwindigkei2value;
 unsigned char Bremskraft3value;
 time_t t1,t2;
 long int  counter,durchlaeufe;
counter=0;
rewind(stdin);
printf("\nNumber of Tests :");
scanf("%ld",&durchlaeufe);
time(&t1);
do{
  counter++;
  Abstand1value=(unsigned char)(rand()*(255-
0.00000000000000E+0000)/(float)RAND_MAX+ 0.00000000000000E+0000);
  printf("\nAbstand1: %u",Abstand1value);
  Geschwindigkei2value=(unsigned char)(rand()*(255-
0.00000000000000E+0000)/(float)RAND_MAX+ 0.00000000000000E+0000);
  printf("\nGeschwindigkei2: %u",Geschwindigkei2value);
  fuzzy(changed,
     Abstand1value,
     Geschwindigkei2value,
    &Bremskraft3value);
  printf("\nAbstand1= %u \t unsigned char ",Abstand1value);
  printf("\nGeschwindigkei2= %u \t unsigned char ",Geschwindigkei2value);
  printf("\nBremskraft3= %u \t unsigned char ",Bremskraft3value);
  }while (counter<durchlaeufe);
time(&t2);
printf("\n Dauer : %f[sec] fuer %d Tests!",difftime(t2,t1),counter);
return 0;
}
```

Generiertes Hauptprogramm zum Testen des C-Codes (hier für den Fall der Zufallswerterzeugung)

Compilieren und Ausführen des C-Codes

Sofern ein C-Compiler zur Verfügung steht, kann der generierte C-Code direkt aus der Entwicklungsumgebung heraus compiliert und - wenn Testcode erzeugt wurde - auch ausgeführt werden. Zum Compilieren werden die Menüfolgen COMPILIEREN | TESTCOMPILAT GENERIEREN bzw. COMPILIEREN | HARDWARE-COMPILAT GENERIEREN herangezogen. Diese Menüfolgen bewirken die Ausführung der über OPTIONEN | CODE-GENERIEREN vorgewählten Kommandozeilenaufrufe der entsprechenden Compiler. Der eventuelle Aufruf der erzeugten EXE-Datei ist im Anschluß daran über COMPILIEREN | AUSFÜHREN möglich.

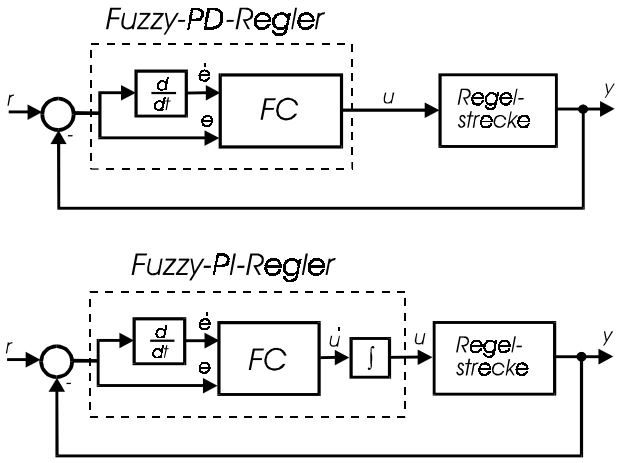


Ausführen des Testcodes
(hier mit manueller Vorgabe scharfer Eingangswerte)

9 Entwurf einfacher Fuzzy-PID-Regler mit FuzzyPID

Leistungsumfang

Das Programm FuzzyPID ermöglicht den interaktiven Entwurf eines Fuzzy-PI- bzw. Fuzzy-PD-Reglers für einen einschleifigen Standardregelkreis mit linearer Regelstrecke. Der geschlossene Regelkreis kann simuliert werden, wobei alle interessierenden Zeitverläufe grafisch dargestellt werden. FuzzyPID ist besonders geeignet für erste "Gehversuche" im Bereich Fuzzy Control sowie Demonstrations- und Lehrzwecke. Komplexere Fuzzy-Regelkreise sollten mit BORIS entworfen werden.



Einschleifiger Regelkreis mit Fuzzy-PD-Regler (oben) bzw. Fuzzy-PI-Regler (unten)

Bedienung und Konzeption des Programms sind im wesentlichen mit der in Kapitel 7 beschriebenen Fuzzy-Shell FLOP identisch. Folgende zusätzliche Einschränkungen sind zu beachten:

- Die linguistischen Variablen tragen die Bezeichnungen
  - *e* für die Regelabweichung *e*,
  - *de/dt* für die zeitliche Änderung $\dot{e}$ der Regelabweichung,
  - *u* für die Stellgröße *u* (Fuzzy-PD-Regler),
  - *du/dt* für die zeitliche Änderung $\dot{u}$ der Stellgröße (Fuzzy-PI-Regler).

Diese Bezeichnungen sind im Programm festgelegt und können nicht geändert werden. Ebensowenig können linguistische Variablen gelöscht werden.

- Die Anzahl der linguistischen Terme pro linguistischer Variable ist auf fünf festgelegt und kann nicht geändert werden. Die Terme tragen für alle Variablen die Bezeichnungen

Negative_Big	(Kürzel --),
Negative_Small	(Kürzel -),
Zero	(Kürzel 0),
Positive_Small	(Kürzel +),
Positive_Big	(Kürzel ++).

  Auch diese Bezeichnungen können nicht geändert werden.

- Der Regler und damit die Regelbasis muß in jedem Fall zwei Eingangsgrößen, nämlich e und $\dot{e}$, aufweisen. Möchte man daher einen Fuzzy-P-Regler realisieren, so wählt man zunächst den Fuzzy-PD-Regler aus und sorgt dann dafür, daß die generierte Stellgröße von $\dot{e}$ unabhängig ist. Dies kann man erreichen, indem man in der Regelmatrix zeilenweise gleiche Einträge wählt.
- Alle Regeln werden grundsätzlich mit dem Wert 1 gewichtet. Negierte Teilprämissen sind nicht möglich.

Da die grundsätzlichen Eingabefunktionen wie die Bearbeitung von Fuzzy Sets und das Erstellen der Regelbasis bereits in Kapitel 7 im Detail erläutert wurden, werden wir uns im Rahmen dieses Kapitels lediglich auf die für dieses Programm spezifischen Operationen beschränken.

Bildschirmaufbau

FuzzyPID zeichnet sich dadurch aus, daß sämtliche relevanten Informationen jederzeit im Hauptfenster des Programms dargestellt werden:

- Die Struktur des zugrundeliegenden Regelkreises

- Die Zugehörigkeitsfunktionen für die Ein- und Ausgangsgrößen des Reglers
- Die aktuelle Regelbasis
- Die Simulationsergebnisse

Das nachfolgende Bild zeigt den Bildschirm unmittelbar nach dem Aufruf des Programms. Man erkennt, daß die Regelbasis zunächst leer ist und die Zugehörigkeitsfunktionen für die linguistischen Terme alle identisch sind. Das Hauptmenü des Programms enthält folgende Optionen, die wir zum größten Teil in den nachfolgenden Abschnitten genauer unter die Lupe nehmen werden:

- DATEI

  Ermöglicht die Ein- und Ausgabe von Datensätzen über eine Datei mit der Extension FUZ. Diese Dateien enthalten alle Zugehörigkeitsfunktionen und die Regelbasis und sind vom Format her zum Programm FLOP kompatibel. Somit können Fuzzy Controller, die mit FuzzyPID entworfen wurden, mit FLOP näher analysiert werden. Auch die umgekehrte Vorgehensweise ist denkbar. Die Unterpunkte dieser Menüoption entsprechen i. a. denen im Modul FLOP.

- REGELKREIS

  Enthält alle Unterpunkte zur Wahl der Regelstrecke, des Reglertyps, der Führungsgröße und der Simulationsparameter.

- VARIABLEN

  Erlaubt die Bearbeitung der linguistischen Variablen und ihrer Terme.

- REGELBASIS

  Ermöglicht die Erstellung und Bearbeitung der Regelbasis.

- SIMULATION

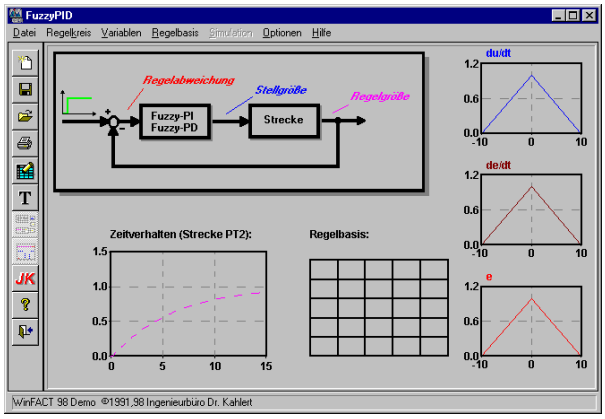
  Startet die Simulation. Dieser Menüpunkt ist erst anwählbar, nachdem eine Regelbasis erstellt bzw. aus einer Datei gelesen wurde.

- OPTIONEN

  Enthält einige Programmoptionen wie Inferenzmechanismus, Defuzzifizierungsmethode und die während der Simulation angezeigten Zeitverläufe.

- HILFE

  Hilfefunktion und Info-Box



Hauptfenster des Programms nach dem Aufruf

Regelkreis

Der dem Programm zugrundeliegende Regelkreis besitzt eine einschleifige Struktur mit Ausgangsgrößenrückführung. Für erste Experimente auf dem Gebiet Fuzzy Control kann die Regelstrecke über die Menüfolge REGELKREIS | REGELSTRECKE aus fünf verschiedenen, fest vorgegebenen Typen ausgewählt werden:

- PT_2-Strecke (nicht schwingfähig)

  Dieser Streckentyp weist eine Streckenverstärkung von 1 und zwei reelle Eigenwerte auf. Die zugehörigen Zeitkonstanten liegen bei $T_1 \approx 0.2$ s und $T_2 \approx 5$ s. Die zugehörige Übertragungsfunktion lautet

$$G(s) = \frac{1}{s^2 + 6s + 1}.$$

- PT_2-Strecke (schwingfähig)

Diese Strecke besitzt zwei konjugiert komplexe Eigenwerte und ist somit schwingfähig. Die Dämpfung liegt bei $\zeta = 0.25$, die Eigenfrequenz bei $\omega_n = 1$. Die zugehörige Übertragungsfunktion lautet

$$G(s) = \frac{1}{s^2 + 0.5s + 1}.$$

- PT_1-I-Strecke

  Reihenschaltung aus einem PT_1-Glied mit einer Zeitkonstanten von $T = 10$ s und einem Integrierglied. Die zugehörige Übertragungsfunktion lautet

$$G(s) = \frac{0.2}{s(1+10s)}.$$

- PT_1-T_t-Strecke

  PT_1-Glied mit Totzeit. Das PT_1-Glied besitzt die Zeitkonstante $T = 5$s. Die Totzeit beträgt $T_t = 3$s. Die zugehörige Übertragungsfunktion lautet somit

$$G(s) = \frac{1}{1+5s} e^{-3s}.$$

- PT_2- Strecke (zeitvariant)

  Entspricht strukturell der schwingfähigen PT_2- Strecke, weist jedoch einen zeitabhängigen Verstärkungsfaktor auf:

$$G(s) = \frac{K(t)}{s^2 + 0.5s + 1}.$$

  Dabei steigt der Verstärkungsfaktor für $0 \le t \le 5$ zunächst linear von 1 auf 2 und fällt danach für $5 \le t \le 10$ wieder linear auf 1 ab:

$$K(t) = \begin{cases} 1 + t/5 & \textit{für } 0 \le t \le 5 \\ 2 - (t-5)/5 & \textit{für } 5 < t \le 10 \\ 1 & \textit{für } t > 10 \end{cases}$$

Eine Modifikation der Streckenparameter ist bei diesen Streckentypen nicht möglich.

Die letzten drei Optionen im Untermenü REGELKREIS | REGELSTRECKE erlauben darüber hinaus die Vorgabe einer beliebigen linearen Regelstrecke in Form einer Übertragungsfunktion. Diese kann über Tastatur oder Datei eingelesen und ebenso wieder in einer Datei vom Typ UFK abgelegt werden.

Als Reglertyp sind Fuzzy-PI- und Fuzzy-PD-Regler zugelassen. Eingangsgrößen des Reglers sind jeweils die Regelabweichung e und ihre zeitliche Änderung $\dot{e}$, Ausgangsgröße ist im Falle des PI-Reglers die zeitliche Änderung $\dot{u}$ der Stellgröße, beim PD-Regler die Stellgröße u selbst. Die Auswahl des Reglertyps wird über die Menüfolge REGELKREIS | REGLERTYP PI *bzw.* REGELKREIS | REGLERTYP PD vorgenommen.

Über die Menüfolge REGELKREIS | SIMULATIONSPARAMETER ... oder die Tastenkombination <Strg><P> kann der Eingabedialog für die Simulationsparameter erreicht werden. Dieser ermöglicht

- die Eingabe der Simulationsdauer,
- die Eingabe der Simulationsschrittzahl,
- die Beeinflussung der Simulationsgeschwindigkeit.

Eine Verringerung der Simulationsgeschwindigkeit kann beispielsweise dann sinnvoll sein, wenn man bei einem sehr schnellen Rechner den Ablauf der Simulation genauer studieren möchte.

Die Führungsgröße $r(t)$ für den Regelkreis kann aus drei verschiedenen Testsignalen ausgewählt werden:

- einer sprungförmigen Anregung mit der Amplitude 1,
- einer rampenförmigen Anregung mit der Steigung

  $$\frac{\Delta r}{\Delta t} = \frac{2}{T_{\text{Ende}}}$$

  und einer maximalen Amplitude von 1, wobei TEnde die Simulationsdauer bezeichnet,
- einem Doppelimpuls der Form

$$r(t) = \begin{cases} 1 & \textit{für} \quad 0 < t \le T_{\text{Ende}} / 3 \\ -1 & \textit{für} \quad T_{\text{Ende}} / 3 < t \le 2T_{\text{Ende}} / 3 \\ 0 & \textit{für} \quad t > 2T_{\text{Ende}} / 3 \end{cases}.$$

Eingabedialog für Simulationsparameter

Simulation

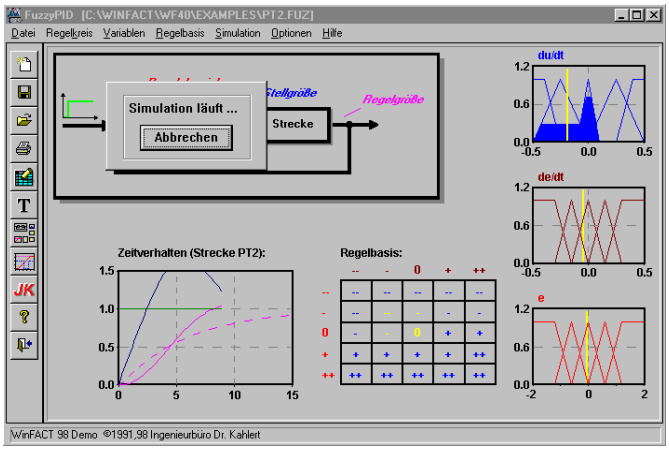
Nachdem Zugehörigkeitsfunktionen und Regelbasis erstellt bzw. aus einer Datei gelesen wurden, kann die Simulation gestartet werden. Dazu stehen zwei unterschiedliche Optionen zur Auswahl, die sich im wesentlichen in der Art der während der Simulation dargestellten Daten unterscheiden.

Über die Menüfolge SIMULATION | ALLES ANZEIGEN oder die Tastenkombination <Strg><S> wird ein Simulationslauf mit Anzeige sämtlicher relevanter Größen gestartet. Der Ablauf der Simulation kann im Hauptfenster des Programms unmittelbar verfolgt werden:

- In den Diagrammen mit den Zugehörigkeitsfunktionen für die Regler-ein- und -ausgangsgrößen am rechten Bildrand werden die aktuellen Werte in Form eines gelben Balkens angezeigt. Das obere Diagramm für die Stellgröße bzw. Stellgrößenänderung (je nach gewähltem Reglertyp) zeigt zusätzlich die aktiven Ausgangs-Fuzzy-Mengen und ihren Erfüllungsgrad an (blau schraffiert).
- In der Regelbasis werden die gerade aktiven Regeln gelb hervorgehoben.
- Das Diagramm in der linken unteren Bildschirmecke stellt schließlich den zeitlichen Verlauf aller Größen dar, wobei sich einzelne Größen über die Hauptmenüoption OPTIONEN ein- bzw. ausschalten lassen:

- Das Zeitverhalten der Strecke allein, d. h. ohne Regler und Rückführung (gestrichelte violette Kurve)
- Die Führungsgröße $r(t)$ des Regelkreises (durchgezogene grüne Kurve)
- Die Regelgröße, d. h. Ausgangsgröße $y(t)$ des Regelkreises (durchgezogene violette Kurve)
- Die Regelabweichung $e(t)$ (durchgezogene hellrote Kurve)
- Die zeitliche Änderung $\dot{e}(t)$ der Regelabweichung (durchgezogene rote Kurve)
- Die Stellgröße $u(t)$ (durchgezogene dunkelblaue Kurve)
- Die zeitliche Änderung $\dot{u}(t)$ der Stellgröße (durchgezogene hellblaue Kurve). Diese kann nur im Falle eines PI-Reglers angezeigt werden.

Die Simulation kann durch Betätigung der Schaltfläche *Abbrechen* jederzeit abgebrochen werden.



Hauptfenster des Programms während der Simulation

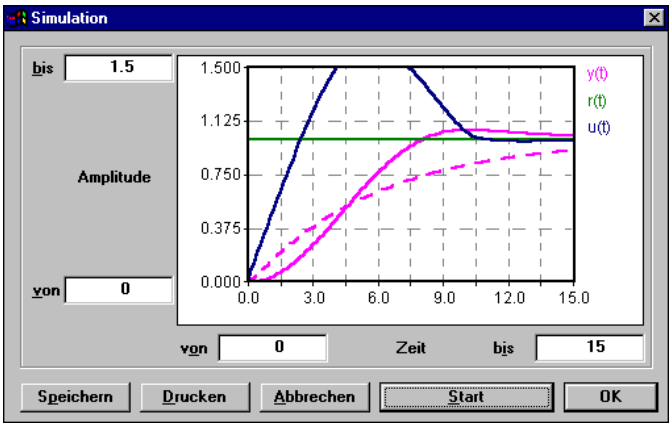
Ist nur das Zeitverhalten des Regelkreises von Interesse, so ist es zweckmäßiger, die Simulation über die Menüfolge SIMULATION | ZEITVERHALTEN bzw. die Tastenkombination <Strg><V> zu starten. Wir gelangen dann in das Dialogfenster wie im folgenden Bild. Hier steht uns für die Darstellung des Zeitverhaltens ein wesentlich größeres Ausgabefenster zur Verfügung und wir haben zu-

dem die Möglichkeit, die Skalierung der Achsen zu beeinflussen.[1] Während das Streckenverhalten unmittelbar nach Aufruf des Dialogfensters ausgegeben wird, können wir die Simulation des Regelkreises über die Schaltfläche *Start* anwerfen. Da die zeitaufwendige Darstellung der Fuzzy Sets und der Regelbasis bei dieser Simulationsart entfällt, läuft der gesamte Vorgang erheblich schneller ab.

Da die zeitliche Änderung $\dot{e}$ der Regelabweichung während der Simulation bei sprungförmigen Führungsgrößenänderungen sehr große Werte annehmen kann, wird sie automatisch auf den vorgegebenen Wertebereich der linguistischen Variable beschränkt.

Das Simulationsdialogfenster enthält zusätzlich folgende Optionen, die über die Schaltflächen am unteren Fensterrand erreichbar sind:

- Über *Drucken* können die Simulationsergebnisse auf dem Standarddrucker ausgegeben werden.
- Über *Speichern* können alle simulierten Größen jeweils in einer Datei vom Typ SIM abgelegt werden. Dazu wird nach Betätigung der Schaltfläche ein Popup-Menü angeboten, aus dem die zu speichernde Größe ausgewählt werden kann.



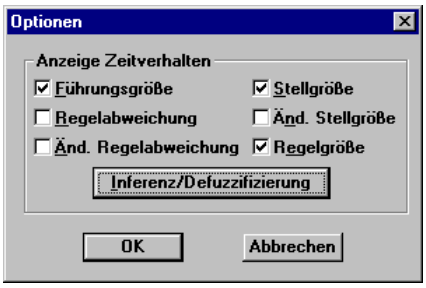
Dialogfenster zur Simulation des Zeitverhaltens

[1] Man beachte, daß eine Umskalierung der Zeitachse lediglich den dargestellten Ausschnitt, nicht aber die Simulationsdauer (Anfangs- bzw. Endzeit der Simulation) beeinflußt!

Optionen

Der Menüpunkt OPTIONEN des Hauptmenüs führt in einen Eingabedialog, der die Änderung folgender Programmparameter ermöglicht:

- Der während der Simulation im Diagramm *Zeitverhalten* dargestellten Größen
- Des Inferenzschemas und der Defuzzifizierungsmethode
- Des Systemverhaltens für den Fall, daß keine der definierten Regeln aktiv ist



Programmoptionen

Beispieldateien

Das Beispielverzeichnis enthält für jeden der möglichen Regelstreckentypen einen Beispieldatensatz mit einem für sprungförmige Führungsgrößen bereits einigermaßen zufriedenstellend arbeitenden Fuzzy Controller. Diese Datensätze können als Ausgangsbasis für eigene Experimente benutzt werden. Die entsprechenden Dateien tragen die Bezeichnungen

PT2.FUZ	Fuzzy-PI-Regler für PT_2-Strecke (nicht schwingfähig),
PT2S.FUZ	Fuzzy-PI-Regler für PT_2-Strecke (schwingfähig),
PT1I.FUZ	Fuzzy-PD-Regler für PT_1-I-Strecke,
PT1TT.FUZ	Fuzzy-PI-Regler für PT_1-T_t-Strecke,
PT2ZV.FUZ	Fuzzy-PI-Regler für PT_2-Strecke (zeitvariant).

Der jeweilige Reglertyp (PI bzw. PD) wird beim Einlesen der Datei anhand der Bezeichnungen für die Eingangsgrößen automatisch erkannt und vom Programm voreingestellt.

10 Blockorientierte Simulation mit BORIS

Übersicht

Anwendungsbereiche von BORIS

Das blockorientierte Simulationssystem BORIS ermöglicht die Simulation nahezu beliebig strukturierter dynamischer Systeme und eignet sich in Verbindung mit der Prozeßschnittstelle und der optionalen C-Code-Generierung damit gleichermaßen für die Anwendungsbereiche

- Meßwerterfassung und Signalanalyse,
- Regelkreisanalyse und -synthese,
- Systemoptimierung,
- Digitaltechnik,
- Konfigurierung und Parametrierung von Hardwarebaugruppen.

Neben den bekannten konventionellen Systemen können insbesondere auch Systeme mit Fuzzy- und/oder Neuro-Komponenten verarbeitet werden.

Einige Leistungsmerkmale von BORIS

Einige der wesentlichen Leistungsmerkmale von BORIS (je nach Ausbaustufe der erworbenen Version):

- Umfangreiche Systembibliothek:
  - Signalgeneratoren (Dreieck, Rechteck, Sinus, Impuls, Rauschen, div. Testfunktionen)
  - Lineare Standardglieder (PT_1, PT_2, ...)
  - Lineare Übertragungsglieder höherer Ordnung
  - Nichtlineare Kennlinienglieder; frei definierbare Kennlinien; algebraische Funktionen
  - Lineare und nichtlineare Standard-Reglerkomponenten

  - Fuzzy Controller und Neuronale Netze
  - Anbindung benutzerdefinierter Funktionsblöcke über DLLs
  - Verschiedene Ausgabeblöcke (Zeitverläufe, Trajektorienverläufe, analoge und digitale Anzeigeinstrumente, Oszillograph, Statusanzeigen)
  - Aktionsblöcke zum interaktiven Eingriff in den Simulationsablauf (z. B. Schalter, Potentiometer)
  - Ein- und Ausgabe von Signalverläufen aus bzw. in Dateien
  - Spektralanalyse über Fast-Fourier-Transformation
  - Statistik-Funktionen
  - Digitale Bausteine (z. B. Logikgatter und Flip-Flops)
  - Kommunikations-Bausteine (z. B. DDE- und TCP/IP-Blöcke)
- Definition hierarchischer Makros (Superblöcke)
- Anzahl der Systemblöcke nur durch den Hauptspeicher des Rechners begrenzt
- Beliebige Plazierbarkeit von Systemblöcken; nahezu beliebig große, scrollfähige Arbeitsfläche
- Verschiedene Integrationsverfahren
- Automatischer oder halbautomatischer Entwurf von PID-Reglern anhand von Einstellregeln
- Numerische Parameteroptimierung auf der Basis von Evolutionsstrategien
- Prozeßankopplung über A/D- und D/A-Wandlerkarten, serielle Meßmodule, Prozeßleitsysteme etc.
- Überführung beliebiger Systemstrukturen in ANSI-C-Code durch den leistungsfähigen AutoCode-Generator

Im nachfolgenden Kapitel soll zunächst anhand eines ausführlichen Beispiels eine Einführung in die Arbeit mit BORIS erfolgen.

Eine erste Simulation mit BORIS

Ein einfaches Beispiel

Die Benutzeroberfläche von BORIS orientiert sich im wesentlichen am Windows-Standard sowie den WinFACT-typischen Konventionen. Im folgenden wollen wir uns in die grundsätzliche Handhabung des Programms anhand eines einfachen Beispiels einarbeiten.

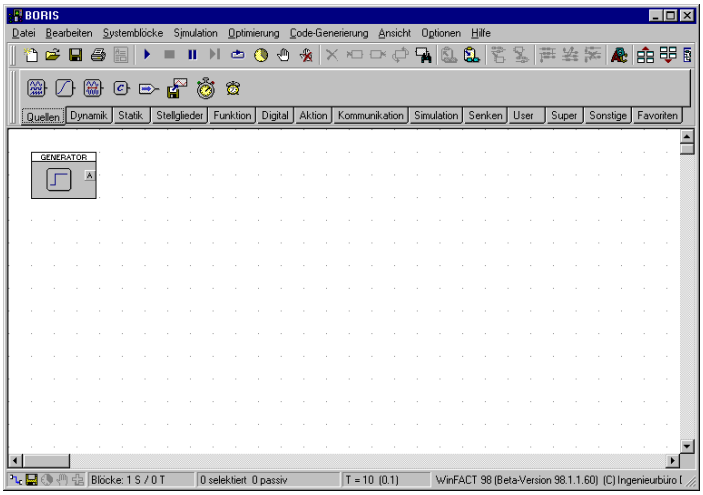
Das im folgenden beschriebene Beispiel befindet sich unter dem Namen DEMO1.BSY im Beispiel-Verzeichnis.

Wir wollen eine lineare Regelstrecke 2. Ordnung mit der Verstärkung $K = 2$ und den Zeitkonstanten $T_1 = 1$ und $T_2 = 2$ betrachten. Uns interessiert zunächst die Antwort $y(t)$ des Systems auf eine sinusförmige Eingangsgröße $u(t) = u_0 \sin \omega t$ mit der Amplitude $u_0 = 1$ und der Kreisfrequenz $\omega = 3$. Wie läßt sich dieses Problem mit BORIS lösen?

Betrachten wir zunächst das Anwendungshauptfenster von BORIS, wie es sich uns direkt nach dem Aufruf des Programms darstellt. Wir erkennen am oberen Rand zwei horizontale Toolbars. Die untere, die sogenannte *Systemblock-Toolbar* ermöglicht den direkten Zugriff auf alle Systemblöcke und ist in verschiedene Blockgruppen (*Palettenseiten*) aufgeteilt, die über die Register aktiviert werden können.

Systemblock-Toolbar von BORIS

Wir benötigen zunächst einen *Signalgenerator* zur Erzeugung unseres Eingangssignals. Diesen erhalten wir durch Anklicken der ersten Schaltfläche der Palettenseiten *Quellen*. Der Generator erscheint daraufhin in der linken oberen Ecke des Zeichenfensters. Das Einfügen des Systemblocks wird weiterhin in der Statuszeile des Hauptfensters quittiert.



Hauptfenster von BORIS nach dem Einfügen des Signalgenerators

Als nächstes wollen wir die Regelstrecke selbst einfügen. Dazu betätigen wir die mit *PT1T2* beschriftete Schaltfläche der Palette *Dynamik*. Der Block erscheint daraufhin rechts neben dem Generator.

Verschieben von Blöcken

Wir wollen den Block noch etwas nach rechts verschieben. Dazu aktivieren wir den Block durch Anklicken mit der linken Maustaste. Die Aktivierung ist am invertierten Blocktitel erkennbar. Bei festgehaltener linker Maustaste können wir den Block jetzt an der gewünschten Position plazieren.

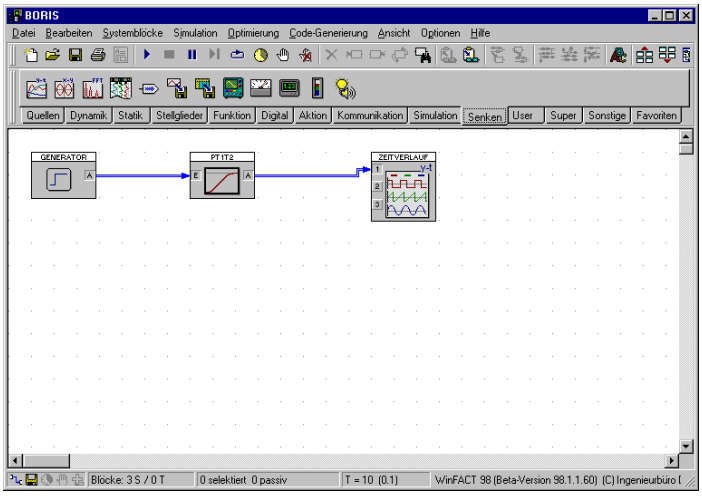
Alternativ lassen sich einzelne Blöcke auch per Drag & Drop einfügen. Dazu wird die linke Maustaste nach dem Anklicken der Block-Schaltfläche festgehalten; der Block kann dann auf dem Arbeitsblatt direkt in die gewünschte Position gebracht und dort „fallengelassen“ werden.

Nun benötigen wir noch einen Ausgabeblock zur späteren Anzeige der Simulationsergebnisse. Wir wollen dazu einen Block vom Typ *Zeitverlauf* wählen, den wir jetzt wieder über die Toolbar (Palettenseite *Senken*) einfügen und im Anschluß daran geeignet plazieren. Gleichzeitig mit der Darstellung des Blocksymbols wird ein zusätzliches Fenster geöffnet, das zunächst zum Symbol verkleinert ist. Dieses Anzeigefenster enthält später die eigentliche Simulationsgrafik.

Als nächstes wollen wir die notwendigen Verbindungen ziehen. Dabei gilt:

*Verbindungen werden **vom Ausgang zum Eingang** gezogen!*

Wir klicken daher zunächst den Ausgang unseres Signalgenerators - das mit "A" beschriftete, erhöht dargestellte Feld - mit der linken Maustaste an. Der Mauszeiger wechselt daraufhin seine Form und stellt - genügend Phantasie beim Anwender vorausgesetzt - einen stilisierten Lötkolben dar. Wir setzen den Cursor danach in das Eingangsfeld "E" unserer Regelstrecke - der Mauszeiger wechselt seine Farbe auf schwarz und zeigt ein Fadenkreuz - und betätigen nochmals die linke Maustaste. Die Verbindung wird daraufhin automatisch eingezeichnet. Auf die gleiche Weise verbinden wir den Ausgang der Strecke mit dem Eingang 1 des Zeitverlauf-Blocks.



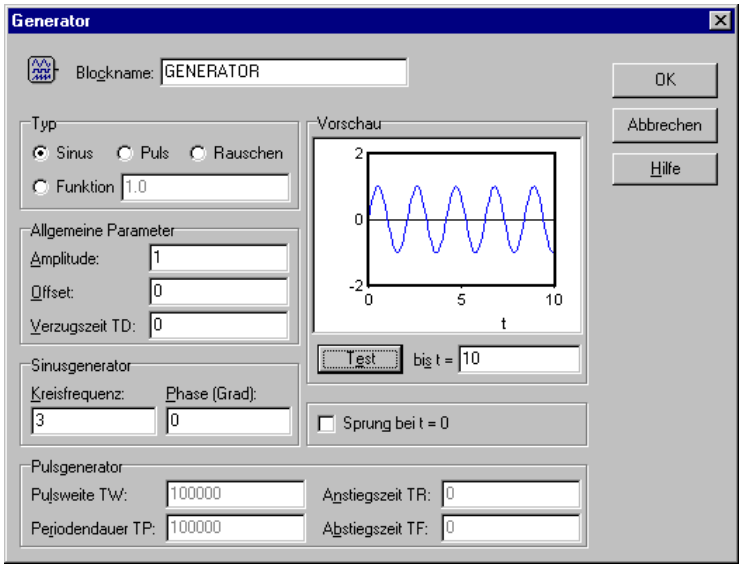
Hauptfenster nach Einfügen aller Blöcke und Ziehen der Verbindungen

Parametrierung der Blöcke

Die Struktur unseres Systems liegt nun fest. Bevor die Simulation gestartet werden kann, müssen wir unsere Blöcke natürlich noch parametrieren. Zunächst wollen wir dies für den Generator tun. Der einfachste Weg dazu besteht in einem Doppelklick mit der linken Maustaste auf den Systemblock. Es erscheint dann der zugehörige Eingabedialog, in dem wir die Änderungen vornehmen können. Wir müssen für unser Beispiel folgende Änderungen vornehmen, die in untenstehendem Dialog bereits eingetragen wurden:

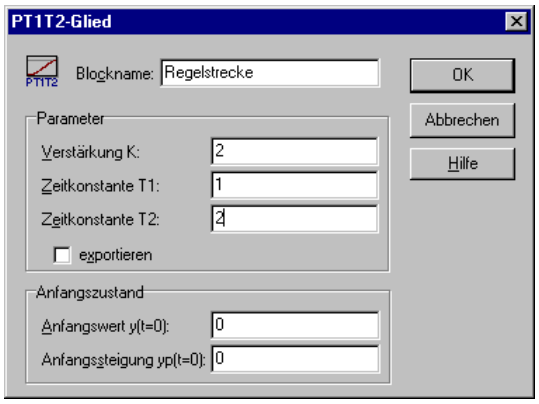
- In der Schaltergruppe *Typ* wählen wir *Sinus* an.
- Im Feld *Kreisfrequenz* tragen wir den Wert 3 ein.

Nach Durchführung der Änderungen können wir die Einstellungen über die Schaltfläche *Test* austesten.

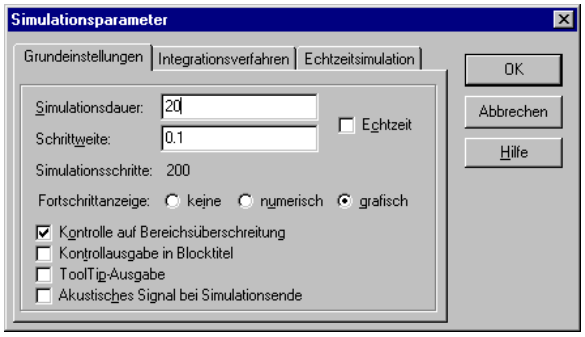


Parameterdialog für Generator

Auf die gleiche Weise parametrieren wir die Regelstrecke. Zur Verdeutlichung ändern wir hier zusätzlich den Blocknamen auf *Regelstrecke*. Zum Abschluß müssen wir die Simulationsparameter festlegen. Dazu wählen wir das Uhrensymbol in der oberen horizontalen Toolbar, der sogenannten *System-Toolbar* und gelangen dadurch in den entsprechenden Eingabedialog. Wir wählen eine Simulationsdauer von 20 und eine Simulationsschrittweite von 0.1. Aus diesen Einstellungen resultieren insgesamt 200 Simulationsschritte.

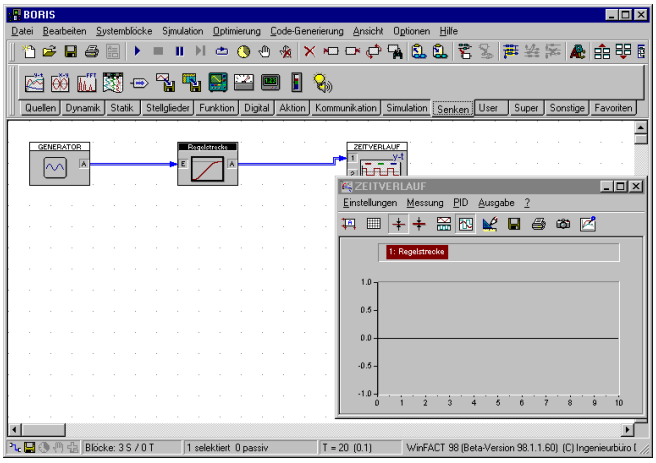


Parametrierung der Regelstrecke



Einstellung der Simulationsparameter

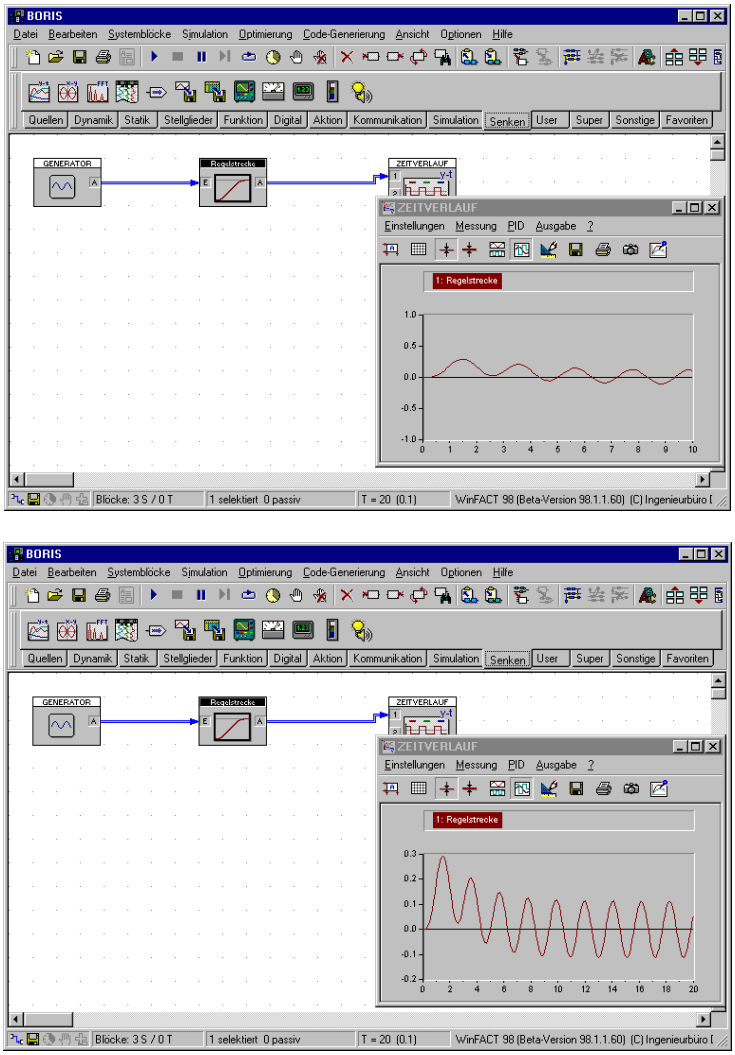
Unser System ist jetzt simulationsbereit. Um den Simulationsverlauf bereits während der Simulation verfolgen zu können, holen wir das Zeitverlauf-Anzeigefenster durch einen Doppelklick aus der Symboldarstellung zurück. Es wird daraufhin in Normalgröße dargestellt, enthält allerdings natürlich noch keine Ergebnisse. Wir können das Fenster jetzt beliebig vergrößern und verkleinern. Da wir die Simulationsdauer auf 20 erhöht haben, müssen wir die Zeitachse anpassen. Dies können wir über die Menüoption EINSTELLUNGEN des Anzeigefensters bewerkstelligen.



Zeitverlauf-Anzeige in Normalgröße

Der Start der Simulation bedarf nunmehr nur noch eines einzigen Tastendrucks auf den Einfachpfeil in der System-Toolbar. Der Fortgang wird im Statusfenster des Hauptfensters und natürlich auch im Zeitverlauf-Anzeigefenster angezeigt. Nach Ende der Simulation stellen wir fest, daß die Skalierung der Amplituden-

werte innerhalb des Zeitverlauf-Anzeigefensters recht ungünstig ist. Dem können wir durch eine Betätigung der linken Schaltfläche in der Toolbar des Fensters abhelfen. Daraufhin erfolgt ein automatisches Neuzeichnen der Kurve mit optimaler Skalierung.



Zeitverlauf-Anzeigefenstervor (oben) und nach der automatischen Umskalierung (unten)

Wir wollen nun - ausgehend vom bestehenden System - zu einem etwas komplexeren Beispiel übergehen. Unsere Regelstrecke soll zu einem geschlossenen Regelkreis ergänzt werden, der neben der Regelstrecke zunächst noch folgende Komponenten enthalten soll:

- einen P-Regler mit der Verstärkung $K_R = 2$
- ein Meßglied in der Rückführung (PT_1-Glied mit $K = 1$, $T = 0.5$)
- einen Subtrahierer für den Soll-Ist-Vergleich

Das im folgenden beschriebene Beispiel befindet sich unter dem Namen DEMO2.BSY im Beispiel-Verzeichnis.


Da wir die bereits vorhandenen Blöcke auch weiterhin benötigen, reicht es aus, die bestehenden Verbindungen zu löschen. Dazu aktivieren wir zunächst die Regelstrecke (durch Einfachklick mit der linken Maustaste) und löschen dann über die entsprechende Schaltfläche der System-Toolbar die Ausgangsverbindung. Direkt im Anschluß daran können wir über die links daneben liegende Schaltfläche auch die Eingangsverbindung zum Generator löschen.

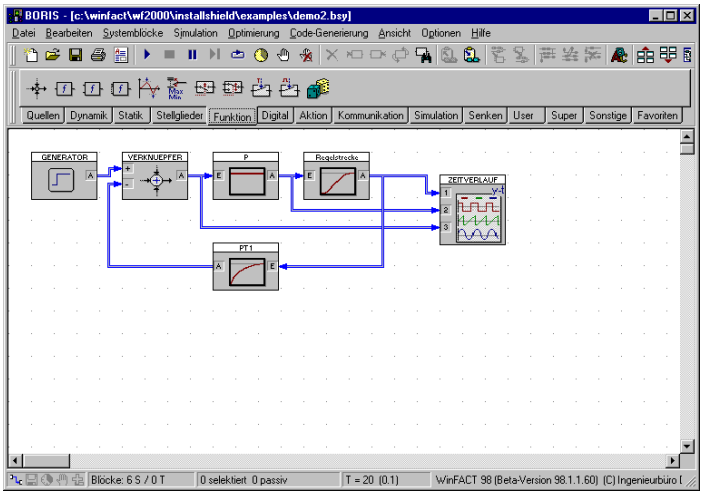
Die folgenden Schritte wollen wir nur noch grob beschreiben. Zunächst fügen wir folgende Blöcke ein:

den P-Regler (Palette *Dynamik*)

das PT_1-Meßglied (Palette *Dynamik*)

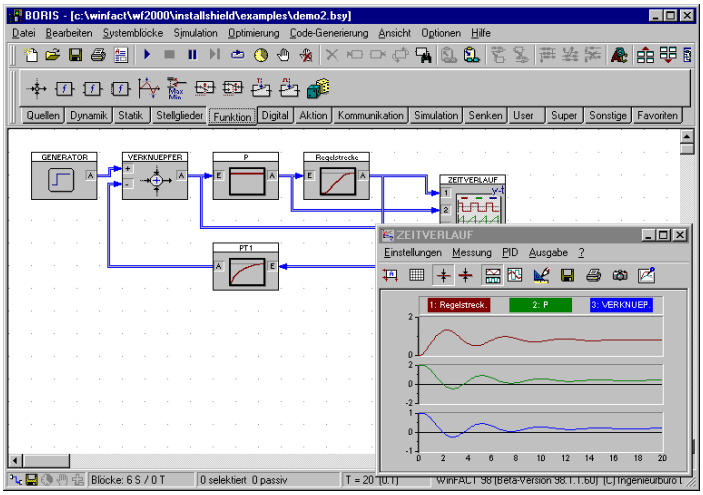
den Subtrahierer (Verknüpfer, Palette *Funktion*)

Im Anschluß daran ziehen wir die entsprechenden Verbindungen und parametrieren die einzelnen Blöcke. Beim Verknüpfer müssen wir insbesondere das negative Vorzeichen für die Rückführung vorsehen. Den Generator schalten wir auf die Betriebsart *Puls* um, so daß er einen Einheitssprung erzeugt. Da das Meßglied in der Rückführung liegt, ist seine Orientierung etwas ungünstig. Dem können wir jedoch auf einfache Weise abhelfen, indem wir den Block um 180° drehen, so daß der Eingang rechts liegt. Die entsprechende Schaltfläche finden wir ebenfalls in der System-Toolbar. An den Zeitverlauf-Block schließen wir jetzt neben der Regelgröße, d. h. dem Ausgang der Regelstrecke, auch die Stellgröße (Ausgang des P-Reglers) und die Regelabweichung (Ausgang des Subtrahierers) an. Sofern wir alle Änderungen korrekt vorgenommen haben, sieht unser Anwendungsfenster nun in etwa wie folgt aus:



Kompletter Regelkreis

Um alle drei darzustellenden Signale in getrennten Diagrammen zu erfassen, deaktivieren wir im über den Menüpunkt EINSTELLUNGEN erreichbaren Dialog des Zeitverlaufs die Option *Alle Kurven in ein Diagramm* bzw. betätigen die entsprechende Toolbar-Schaltfläche. Die anschließende Simulation liefert folgende Ergebnisse:



Simulationsergebnisse

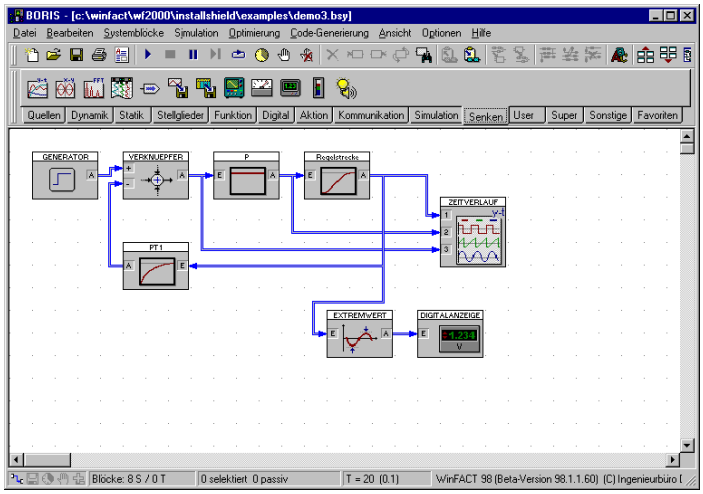
Wir wollen unser System ein letztes Mal erweitern, so daß wir einen genauen Wert für die maximale Ausgangsgröße erhalten. Dazu fügen wir zwei weitere Systemblöcke ein:

einen Extremwert-Block (Palette *Funktion*)

eine Digitalanzeige (Palette *Senken*)

Dieses Beispiel befindet sich unter dem Namen DEMO3.BSY im Beispielverzeichnis.

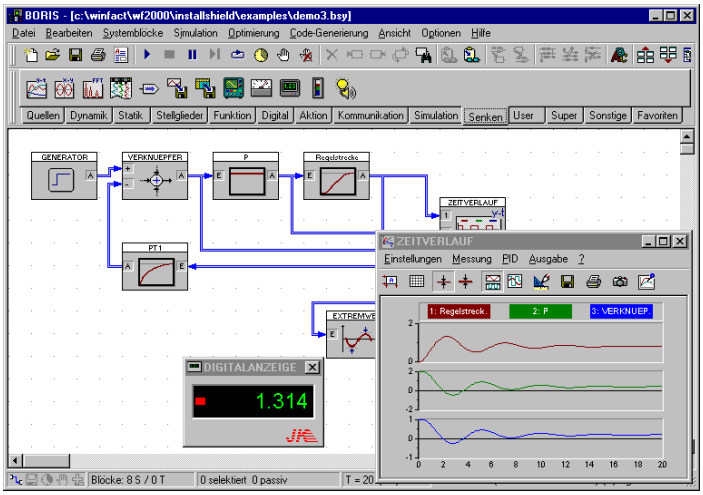
Unser Simulationssystem weist damit folgende Struktur auf:



System mit zusätzlicher Extremwertbestimmung

Vor der Simulation holen wir nun beide Anzeigefenster (Zeitverlauf und Digitalanzeige) aus der Symboldarstellung zurück. Nachfolgende Bildschirmgrafik zeigt das auf diese Weise erhaltene Simulationsergebnis. Dem Digitalinstrument können wir eine maximale Ausgangsgröße von ungefähr 1.31 entnehmen.

Wir wollen die einführenden Beispiele damit abschließen. Die folgenden Kapitel enthalten eine vollständige, detaillierte Beschreibung aller Leistungsmerkmale, Bedienoptionen und zur Verfügung stehenden Systemblöcke.



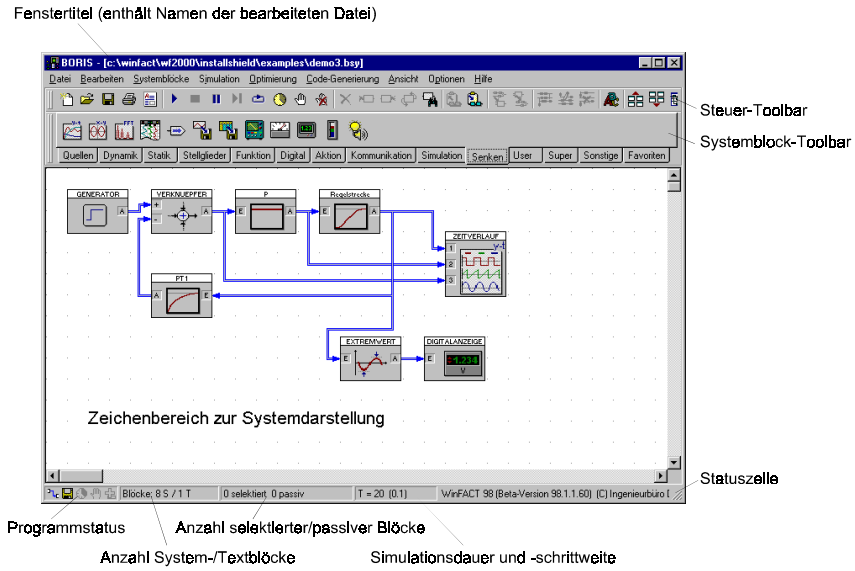
Simulationsergebnis

Aufbau der Simulationsstruktur

Komponenten des BORIS - Hauptfensters

Nachfolgende Grafik zeigt das BORIS-Hauptfenster mit seinen einzelnen Komponenten. Das Fenster enthält neben den Windows - Standardkomponenten die folgenden Bestandteile:

- Eine erste horizontale Toolbar (System-Toolbar) unterhalb des Menüs. Diese Toolbar enthält Schaltflächen für die am häufigsten benutzten Befehle. Über OPTIONEN | TOOLBARS | SYSTEMTOOLBAR SICHTBAR kann die Toolbar wechselweise zu- oder abgeschaltet werden.



- Eine zweite, in mehrere Paletten aufgeteilte horizontale Toolbar (Systemblock-Toolbar). Sie ermöglicht den direkten Zugriff auf alle Systemblöcke. Die Palette *Favoriten* kann vom Anwender beliebig konfiguriert werden (siehe Abschnitt *Konfigurierung der Systemblock-Toolbar*). Über OPTIONEN | TOOLBARS | SYSTEMBLOCKTOOLBAR SICHTBAR kann die Toolbar wechselweise zu- oder abgeschaltet werden.

- Eine Statuszeile am unteren Fensterrand. Sie gibt die Anzahl der aktuell vorhandenen Systemblöcke, die Anzahl der selektierten bzw. passiv gesetzten Blöcke[1] sowie die aktuellen Simulationsparameter in der Form T = T_{Simu} (ΔT) an. Dabei ist T_{Simu} die Simulationsdauer und ΔT die Simulationsschrittweite.

  Während der Simulation zeigt die Statuszeile den Fortlauf der Simulation an, sofern diese Option nicht deaktiviert wurde. Am linken Rand der Statuszeile zeigen fünf Icons den aktuellen Systemzustand an:

   Autorouter aktiv

   System wurde seit der letzten Änderung noch nicht gespeichert

[1] Man beachte, daß bei der Anzeige der Systemblockanzahl nur "echte" Systemblöcke (also keine Textblöcke) gezählt werden, während bei der Anzahl der selektierten bzw. passiven Blöcke alle Blocktypen berücksichtigt werden!

 Simulation läuft

 Breakpoint aktiv

 Optimierung läuft

- Das eigentliche Zeichenfenster zur Anzeige der Systemstruktur. Es ist über seine Bildlaufleisten sowohl in vertikaler als auch in horizontaler Richtung scrollbar. Zu Beginn befindet sich der sichtbare Ausschnitt in der linken oberen Ecke. Dieser Ausgangszustand kann jederzeit über OPTIONEN | BILDAUSSCHNITT IN URSPRUNG wiederhergestellt werden. Das Zeichenfenster weist standardmäßig ein Punktraster auf, an dem alle Systemblöcke mit der linken oberen Ecke ausgerichtet werden. Über die Menüfolge OPTIONEN | AN RASTER AUSRICHTEN läßt sich das automatische Ausrichten deaktivieren; die Systemblöcke sind dann beliebig plazierbar. Die Anzeige des Rasters selbst läßt sich über OPTIONEN | RASTER ANZEIGEN ausschalten.

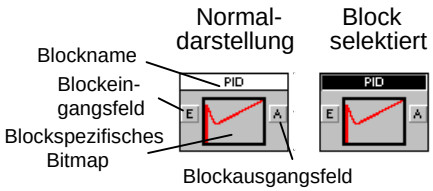
Einfügen und Bearbeiten von Systemblöcken

Das Einfügen und Bearbeiten von Systemblöcken umfaßt folgende Möglichkeiten:

- *Einfügen* neuer Blöcke aus der Systemblock-Bibliothek
- *Selektieren* einzelner Blöcke oder Blockgruppen
- *Verschieben* einzelner Blöcke oder Blockgruppen
- *Löschen* einzelner Blöcke oder Blockgruppen
- *Drehen* einzelner Blöcke
- *Kopieren* und *Einfügen* einzelner Blöcke oder Blockgruppen
- *Deaktivieren* (Passivsetzen) einzelner Blöcke oder Blockgruppen
- *Parametrieren* einzelner Blöcke
- Ändern der *Blockgröße*

Wie ein Block aufgebaut ist

Folgende Grafik zeigt den Aufbau eines Systemblocks:



Er besteht aus folgenden Komponenten:

- Dem *Titelbalken*, der den Blocknamen enthält. Dieser entspricht per Voreinstellung dem Namen des Blocktypen, kann aber vom Anwender über den Parameterdialog modifiziert werden (max. 25 Zeichen). Im selektierten Zustand wird der Titelbalken invertiert. Ist der Blockname zu lang, um vollständig angezeigt zu werden, wird er automatisch abgekürzt.
- Den *Ein-* und *Ausgangsgrößenfeldern*, die zum Ziehen von Verbindungen benötigt werden. Bei Blöcken mit nur einem Ein- und Ausgang ist das Eingangsfeld mit "E" und das Ausgangsfeld mit "A" gekennzeichnet; bei mehreren Ein- oder Ausgängen sind diese in der Regel durchnumeriert oder tragen spezielle Bezeichnungen (z. B. "R" und "S" beim RS-Flip-Flop).
- Einem *blockspezifischen Bitmap*, das den jeweiligen Blocktyp unmittelbar erkennen läßt.

Ein Block kann je nach Blocktyp bis zu fünfzig Ein- und Ausgänge besitzen.

Um einen neuen Block einzufügen...

...gibt es drei verschiedene Möglichkeiten:

1. Sie klicken mit der linken Maustaste die entsprechende Schaltfläche in der Systemblock-Toolbar an. Der Block wird dann automatisch an einer freien Stelle auf dem Arbeitsblatt plaziert.
2. Sie wählen den entsprechenden Blocktyp aus dem SYSTEMBLÖCKE-Untermenü des Hauptmenüs. Auch in diesem Fall wird der Block automatisch plaziert.
3. Sie klicken mit der linken Maustaste die entsprechende Schaltfläche in der Systemblock-Toolbar an, halten die Maustaste gedrückt und ziehen den Block per Drag & Drop an den gewünschten Ort. Beim Loslassen der Maustaste wird der Block dann dort plaziert.

Letztere Vorgehensweise empfiehlt sich vor allem dann, wenn bereits eine Vielzahl von Blöcken auf dem Arbeitsblatt plaziert wurde.

Wie Sie Blöcke selektieren

Zur Selektion einzelner Blöcke oder ganzer Blockgruppen gibt es verschiedene Möglichkeiten:

- Einen einzelnen Block selektieren Sie durch einfaches Anklicken mit der Maus. War zuvor ein anderer Block selektiert, so wird dessen Selektion zurückgenommen.
- Sollen zusätzlich weitere Blöcke selektiert werden, so erreichen Sie dies, indem Sie diese bei festgehaltener <Shift>- oder <Strg>-Taste anklicken. Ein nochmaliges Anklicken eines bereits selektierten Blocks nimmt die Selektion wieder zurück.
- Alternativ dazu können Sie ganze Blockgruppen durch Aufziehen eines Rechtecks mit festgehaltener linker Maustaste selektieren. Alle Blöcke, die vollständig im Rechteck liegen, werden dabei selektiert.
- Über die Menüoption BEARBEITEN | ALLE BLÖCKE SELEKTIEREN lassen sich alle Blöcke gleichzeitig selektieren.

Das Anklicken einer beliebigen freien Stelle innerhalb des Zeichenbereichs nimmt alle Selektionen zurück.

So verschieben Sie Blöcke

Zum Verschieben eines Blocks oder mehrerer Blöcke gehen Sie wie folgt vor:

1. Selektieren Sie zunächst den zu verschiebenden Block bzw. die zu verschiebenden Blöcke.
2. Klicken Sie mit der linken Maustaste den Innenbereich des zu verschiebenden Blocks bzw. - bei der Verschiebung einer Blockgruppe - den Innenbereich eines beliebigen selektierten Blocks bei festgehaltener Maustaste und verschieben Sie die Blöcke in gewünschter Weise. Der Cursor wechselt dabei zur Kreuzform. Bei Erreichen des Fensterrands erfolgt automatisches Scrollen.

Nach der Verschiebung werden die Blöcke gegebenenfalls so ausgerichtet, daß sie sich nicht überlappen.

Verschieben des Gesamtsystems

Während der Systemkonfigurierung kann speziell bei komplexeren Systemen der Fall auftreten, daß links oder oberhalb des Systems weitere Systemblöcke eingefügt werden sollen, dort aber kein Platz mehr zur Verfügung steht. Daher bietet BORIS die Möglichkeit, das Gesamtsystem (d. h. alle Blöcke und Verbindungen) auf einfache Weise zu verschieben, ohne daß diese zuvor selektiert werden müssen. Die Verschiebung kann in alle Richtungen vorgenommen werden und erfolgt bei jedem Schritt um eine Rastereinheit. Die entsprechenden Menübefehle lauten:

OPTIONEN | SYSTEM NACH RECHTS VERSCHIEBEN

OPTIONEN | SYSTEM NACH LINKS VERSCHIEBEN

OPTIONEN | SYSTEM NACH UNTEN VERSCHIEBEN

OPTIONEN | SYSTEM NACH OBEN VERSCHIEBEN

Löschen von Blöcken

Das Löschen von Blöcken kann auf drei Arten erfolgen:

- Durch Selektieren der zu löschenden Blöcke und Wahl der Menüoption BEARBEITEN | BLOCK LÖSCHEN
- Durch Selektieren der zu löschenden Blöcke und Betätigung der Schaltfläche  der System-Toolbar
- Durch Selektieren der zu löschenden Blöcke und einen rechten Mausklick. Im daraufhin erscheinenden Popup-Menü wählen Sie die Option LÖSCHEN. Soll nur ein einzelner Block gelöscht werden, kann dieser direkt mit der rechten Maustaste angeklickt werden, ohne daß er zuvor selektiert werden muß.

Eingangs- und Ausgangsverbindungen der gelöschten Blöcke werden automatisch mitgelöscht.

So lassen sich Blöcke drehen

Die Orientierung eines Blocks kann um 180° gedreht werden, so daß die Eingänge auf der rechten Seite liegen (beispielsweise für Blöcke in einer Rückführung). Dazu wird der Block zunächst selektiert und dann über BEARBEITEN | BLOCK DREHEN oder Betätigung der Schaltfläche  gedreht. Alternativ dazu können Sie den Block mit der rechten Maustaste anklicken und im darauf-

hin erscheinenden Popup-Menü die Option DREHEN wählen. Die Blockverbindungen werden automatisch nachgeführt (dies gilt allerdings nicht für manuelle Verbindungen; siehe dazu später!).

Kopieren und Einfügen

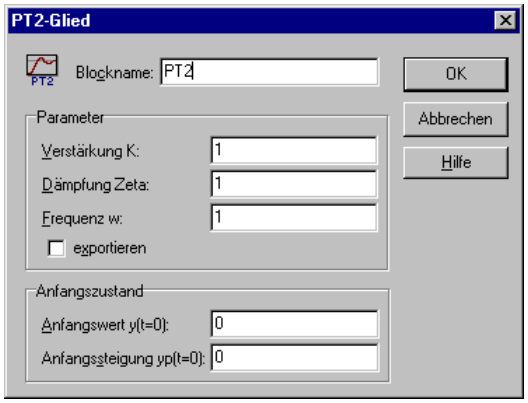
Das Kopieren und Einfügen einzelner Blöcke oder ganzer Blockgruppen läuft über eine temporäre Datei mit Namen BOCOPY.TMP ab. Dabei werden die selektierten Blöcke, ihre Parameter und die Verbindungen der Blöcke untereinander kopiert bzw. eingefügt. Gehen Sie wie folgt vor:

- Selektieren Sie die zu kopierenden Blöcke.
- Wählen Sie die Menüoption BEARBEITEN | KOPIEREN bzw. betätigen die Schaltfläche der System-Toolbar.
- Zum Einfügen wählen Sie BEARBEITEN | EINFÜGEN bzw. die Schaltfläche . Der Cursor wechselt daraufhin seine Gestalt und Sie können die gewünschte Zielposition (linke obere Ecke) des einzufügenden Teilsystems durch Anklicken des Arbeitsblattes vorgeben.

Wie Sie Blöcken ihre Parameter verpassen

Die Parametrierung eines Systemblocks erfolgt am einfachsten durch einen Doppelklick mit der linken Maustaste innerhalb des Blocks. Alternativ dazu kann nach Selektion des Blocks die Menüfolge BEARBEITEN | BLOCK BEARBEITEN aufgerufen werden. Es erscheint dann der blockspezifische Parameterdialog, über den die gewünschten Parameteränderungen vorgenommen werden können. Folgende Grafik zeigt beispielhaft den Parameterdialog für ein schwingfähiges PT_2-Glied.

Der Parameterdialog weist - unabhängig vom Blocktyp - am oberen Rand ein Texteingabefeld für den Blocknamen auf, der in der Blockdarstellung jeweils im Titelbalken erscheint. Dieser Blockname entspricht in der Voreinstellung dem Namen des Blocktyps, kann vom Anwender aber beliebig geändert werden. Die Länge des Blocknamens darf jedoch 25 Zeichen nicht überschreiten. Darüber hinaus weist jeder Parameterdialog eine *Hilfe*-Schaltfläche auf, über die Informationen zum jeweiligen Blocktyp angefordert werden können.



Parameterdialog für schwingfähiges PT_2-Glied

Sein oder nicht sein? So setzen Sie Blöcke passiv

Häufig möchte man gewisse Teile der Simulationsstruktur kurzzeitig "ausschalten", z. B. um verschiedene Modellansätze vergleichen zu können, um File-Output-Blöcke zu deaktivieren oder um bei sehr komplexen Strukturen auch einfach nur Rechenzeit zu sparen. Statt diese Komponenten mühsam aus der Struktur entfernen und später wieder einfügen zu müssen, bietet Ihnen BORIS die Möglichkeit, Blöcke auf Mausklick *passiv zu setzen*. Passiv gesetzte Blöcke werden während der Simulation so behandelt, als wären sie nicht vorhanden. Blockeingänge, die von einem passiv gesetzten Block gespeist werden, werden also wie offene Eingänge behandelt (ein passiver Block in einer Reihenschaltung "unterbricht" die Verbindung also beispielsweise!).

Um einen einzelnen Block passiv zu setzen,

- klicken Sie den Block mit der *rechten* Maustaste an und wählen im daraufhin eingeblendeten Popup-Menü die Option PASSIV

oder (und das ist der umständliche Weg)

- selektieren Sie den Block und wählen im Hauptmenü die Menüoption BEARBEITEN | BLOCK PASSIV.

Wollen Sie mehrere Blöcke gleichzeitig passiv setzen, so

- selektieren Sie die Blöcke,
- klicken mit der rechten Maustaste auf das Arbeitsblatt und wählen im daraufhin eingeblendeten Popup-Menü die Option PASSIV

oder (wiederum umständlicher)

➢ selektieren Sie die Blöcke und wählen im Hauptmenü die Menüoption BEARBEITEN | BLOCK PASSIV.

Um einen passiven Block wieder zu aktivieren,

➢ klicken Sie den Block mit der *rechten* Maustaste an und wählen im daraufhin eingeblendeten Popup-Menü die Option AKTIV

oder (und das ist der umständliche Weg)

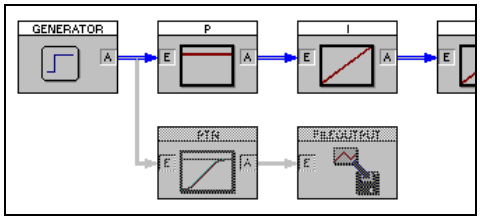
➢ selektieren Sie den Block und wählen im Hauptmenü die Menüoption BEARBEITEN | BLOCK AKTIV.

Um mehrere passive Blöcke zu aktivieren, verfahren Sie sinngemäß.

Um *alle* Blöcke zu aktivieren,

➢ wählen Sie die Menüoption BEARBEITEN | ALLE BLÖCKE AKTIVIEREN.

Passiv gesetzte Blöcke erkennen Sie am grauschattierten Block-Bitmap. Ebenso werden Verbindungen, die von passiven Blöcken ausgehen oder solche speisen, grau dargestellt, sofern diese Option nicht über den Anzeige-Dialog (OPTIONEN | ANPASSEN...) deaktiviert wurde.

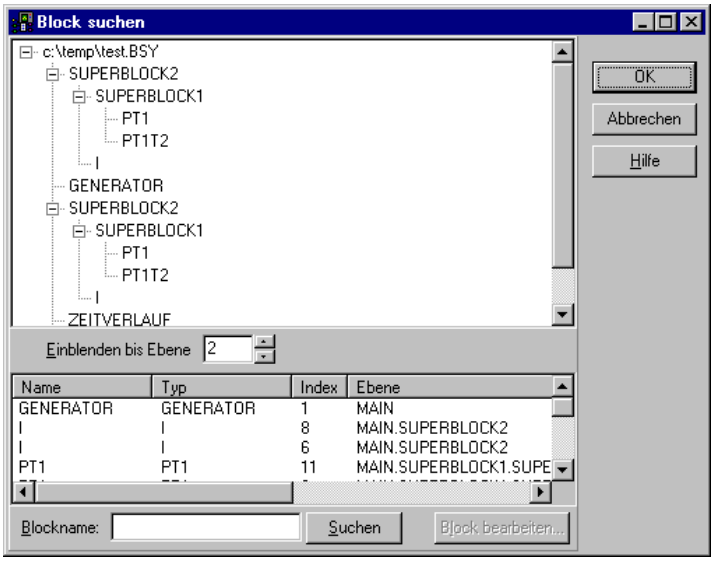


Passivsetzen von Blöcken: Der untere vom Generator ausgehende Zweig wurde hier deaktiviert.

Wo ist er denn? Blöcke suchen (und finden)

Beim Bearbeiten komplexer Simulationsstrukturen ist oft ein schneller Zugriff auf einen bestimmten Block notwendig, z. B. um seine Parameter zu ändern. Das Durchsuchen des Systems über die Bildlaufleisten ist hier oft sehr mühsam; außerdem wird der gesuchte Block in vielen Fällen gar nicht im aktuell geladenen System, sondern z. B. in einem der definierten Superblöcke (u. U. sogar in einem Superblock innerhalb eines Superblocks...) sein. BORIS bietet daher eine leistungsfähige Suchfunktion, die über BEARBEITEN | BLOCK SUCHEN... bzw. die Schaltfläche der System-Toolbar aufgerufen wird.

Der Suchdialog listet nach dem Aufruf im oberen Teil, der Baumstrukturansicht, zunächst alle Blöcke bis zur untersten Hierarchieebene auf. Über das Eingabefeld *Einblenden bis Ebene* können die Ebenen schrittweise ausgeblendet werden. Die untere Ansicht (Listenstruktur) enthält grundsätzlich alle Ebenen. Dabei wird die oberste Ebene (d. h. die aktuell geladene System- oder Superblockdatei) mit *MAIN* bezeichnet, untergeordnete Ebenen werden jeweils durch einen Punkt voneinander getrennt. So bezeichnet z. B. *MAIN.TASTERA* einen Block, der sich in der aktuellen Systemdatei im Superblock *TASTERA* befindet.



Der Block suchen-Dialog von BORIS

Alle Einträge der Listenstruktur sind alphabetisch bezüglich des Blocknamens geordnet. Durch Anklicken einer der Listenüberschriften kann die Sortierung jedoch auch nach einem anderen Kriterium (z. B. dem Blocktyp) erfolgen. Zum Auffinden eines gesuchten Blocks gibt es zwei Möglichkeiten: einen Doppelklick auf die entsprechende Baum- oder Listenansicht oder die Eingabe des Blocknamens (oder eines Teils davon, z. B. *GEN*) im Eingabefeld *Blockname* und Betätigung der *Suchen*-Schaltfläche. Das weitere Verhalten von BORIS hängt nun vom gewählten Blocktyp ab:

- Ein Block *der obersten Hierachieebene* wird automatisch selektiert und der Bildausschnitt - soweit möglich - so verschoben, daß der gesuchte Block genau in der Fenstermitte plaziert ist.

- Wurde ein Block gewählt, der sich *in einem Superblock* befindet (gleichgültig in welcher Hierarchieebene), so wird der entsprechende Superblock automatisch zur Bearbeitung geladen. Der gesuchte Block wird in diesem Fall allerdings *nicht* automatisch mittig plaziert, so daß ggf. innerhalb der geladenen Superblock-Datei erneut die Suchfunktion aufgerufen werden muß!

Blöcke der obersten Hierarchieebene (mit Ausnahme von Superblöcken) können außerdem direkt aus dem Dialog heraus über die Schaltfläche *Block bearbeiten* bearbeitet werden.

Ändern der Blockgröße

BORIS kann alle Systemblöcke in unterschiedlichen Größenstufen darstellen. Ein neu eingefügter Block wird in der Regel in der größten Stufe (100%) dargestellt. Diese Standard-Einstellung kann aber über den Anzeige-Dialog (erreichbar über OPTIONEN | ANPASSEN..., Eingabefeld *Standard-Blockgröße*) geändert werden.

Um die Blockgröße einzelner oder mehrerer Blöcke zu ändern, gehen Sie wie folgt vor:

1. Öffnen Sie das Blockgrößen-Fenster über ANSICHT | BLOCKGRÖßEN-FENSTER.
2. Selektieren Sie die zu modifizierenden Blöcke.
3. Geben Sie im Eingabefeld *Relative Größe* des Blockgrößen-Fensters die gewünschte Größe an und betätigen Sie die *Zuweisen*-Schaltfläche.

Das Blockgrößen-Fenster

Verbinden der Systemblöcke

So verbinden Sie zwei Blöcke miteinander

Alle Verbindungen zwischen Blöcken werden mausgesteuert gezogen. Da ein integrierter Autorouter automatisch für rechtwinklige, möglichst kreuzungsfreie Verbindungen sorgt, müssen in der Regel lediglich die beiden zu verbindenden Blöcke angewählt werden. Bei Bedarf kann der Autorouter über OPTIONEN | AUTOROUTER auch deaktiviert werden, so daß die Verbindungen auch manuell gelegt werden können. Der aktuelle Status wird in der Statuszeile grafisch angezeigt.

Um eine automatische Verbindung zu ziehen, gehen Sie wie folgt vor:

1. Zunächst ist das Ausgangsfeld des Blocks, von dem die gewünschte Verbindung ausgehen soll, mit der linken Maustaste anzuklicken. Der Mauszeiger wechselt dadurch seine Form und stellt nunmehr einen stilisierten Lötkolben dar. Ein *A* neben dem Cursor weist auf den aktivierten Autorouter hin.

2. Jetzt kann das Eingangsfeld des Zielblocks angewählt und durch einen Mausklick mit der linken Taste bestätigt werden. Sobald Sie sich über einem zulässigen Eingangsfeld befinden, wechselt der Cursor seine Farbe auf schwarz und es erscheint zusätzlich ein stilisiertes Fadenkreuz. Während der Mausbewegung wird vom Ausgangsblock ein "Gummiband" nachgeführt. Bei Erreichen des Zeichenfensterrands erfolgt ein automatisches Scrollen. Soll eine begonnene Verbindung rückgängig gemacht werden, so erreicht man dies durch Anklicken einer beliebigen freien Stelle innerhalb des Zeichenfensters.

Ziehen einer automatischen Verbindung

Nach regulärer Beendigung der Verbindung wird diese automatisch mit entsprechender Bepfeilung eingezeichnet. Die Verbindungen werden so gelegt, daß möglichst keine anderen Systemblöcke geschnitten werden. Sollte dies dennoch einmal vorkommen, kann man in den meisten Fällen durch leichtes Verschieben einzelner Blöcke den gewünschten Zustand herstellen. Verbindungen, die von demselben Blockausgang stammen, werden vom Autorouter in der Regel automatisch zusammengefaßt.

Manuelle Verbindungen werden komplett vom Anwender gezogen und können prinzipiell beliebigen Verlauf haben. Zum Ziehen manueller Verbindungen muß der Autorouter ausgeschaltet sein. Um eine manuelle Verbindung zu ziehen, gehen Sie wie folgt vor:

Ziehen einer manuellen Verbindung

1. Klicken Sie das Ausgangsfeld des Blocks, bei dem die Verbindung starten soll, mit der linken Maustaste an.
2. Um einen Knickpunkt anzulegen, betätigen Sie die linke Maustaste. Eine Verbindung kann bis zu 8 Knickpunkte besitzen. Sofern die Option *An Raster ausrichten* aktiviert ist, werden die einzelnen Teilstücke automatisch rechtwinklig ausgerichtet, sofern sie eine bestimmte Neigung nicht überschreiten.
3. Um eine manuelle Verbindung abzuschließen, klicken Sie das Eingangsfeld des Blocks an, in dem die Verbindung enden soll.

Eine begonnene Verbindung können Sie durch *Doppel*klick auf eine freie Stelle des Arbeitsblattes jederzeit zurücknehmen.

Bearbeiten einer manuellen Verbindung

Manuelle Verbindungen können jederzeit modifiziert werden. Dazu gehen Sie wie folgt vor:

1. Klicken Sie die Verbindung mit der linken Maustaste an. Die Knickpunkte werden daraufhin markiert.
2. Die einzelnen Knickpunkte lassen sich nun bei festgehaltener linker Maustaste beliebig verschieben. Wird dabei zusätzlich die <Shift>- oder <Strg>-Taste gedrückt, werden die verschobenen Punkte automatisch auf ein virtuelles 5-Pixel-Raster ausgerichtet.
3. Um einen neuen Knickpunkt einzufügen, klicken Sie die Stelle, an der der neue Punkt eingefügt werden soll, doppelt an.

Umwandeln von Verbindungen

Manuelle Verbindungen können jederzeit in automatische Verbindungen umgewandelt werden und umgekehrt. Um eine manuelle Verbindung in eine automatische Verbindung (bzw. umgekehrt) zu überführen, gehen Sie wie folgt vor:

1. Klicken Sie mit der rechten Maustaste auf das Blockeingangs- bzw. -ausgangsfeld, in das die gewünschte Verbindung läuft.
2. Wählen Sie im daraufhin erscheinenden Popup-Menü die Option VERBINDUNG MANUELL bzw. VERBINDUNG AUTO.

Wollen Sie alle Verbindungen eines Blocks gleichzeitig modifizieren, so erreichen Sie dies wie folgt:

1. Selektieren Sie den Block.
2. Wählen Sie im Hauptmenü die Option BEARBEITEN | VERBINDUNGEN | EINGANGSVERBINDUNGEN MANUELL bzw. entsprechende.

Vom Ausgang eines Blocks können beliebig viele Verbindungen ausgehen. Jeder Blockeingang kann jedoch nur eine Verbindung erhalten. Die Verbindungen können auf verschiedene Weise dargestellt werden. Die entsprechenden Möglichkeiten werden über OPTIONEN | ANPASSEN angeboten.

Wie Sie Verbindungen löschen

Um eine einzelne Verbindung zu löschen, gehen Sie wie folgt vor:

1. Klicken Sie mit der rechten Maustaste auf das Blockeingangs- bzw. -ausgangsfeld, in das die gewünschte Verbindung läuft.
2. Wählen Sie im daraufhin erscheinenden Popup-Menü die Option LÖSCHEN.

Wollen Sie alle Verbindungen eines Blocks gleichzeitig löschen, so erreichen Sie dies wie folgt:

1. Selektieren Sie den Block.
2. Wählen Sie im Hauptmenü die Option BEARBEITEN | VERBINDUNGEN | EINGANGSVERBINDUNGEN LÖSCHEN bzw. BEARBEITEN | VERBINDUNGEN | AUSGANGSVERBINDUNGEN LÖSCHEN oder betätigen Sie in der System-Toolbar die Schaltfläche bzw. .

Jetzt wird`s bunt: Mehrfarbige Verbindungen

Manchmal ist es aus Gründen der Übersichtlichkeit erwünscht, einzelnen Verbindungen unterschiedliche Farben zu geben. Daher erlaubt BORIS für jede Verbindung eine individuelle Farbe. Die Farbe für neu eingefügte Verbindungen ist standardmäßig blau. Diese Einstellung kann aber über den Anzeige-Dialog (erreichbar über OPTIONEN | ANPASSEN...) verändert werden. Um die Farbe bereits vorhandener Verbindungen zu ändern, gehen Sie wie folgt vor. Um die Farbe einer einzelnen Ein- bzw. Ausgangsverbindung eines Blocks zu ändern,

- klicken Sie mit der rechten Maustaste auf das Blockeingangs- bzw. -ausgangsfeld, in das die gewünschte Verbindung läuft und
- wählen im daraufhin erscheinenden Popup-Menü die Option VERBINDUNGSFARBE....

Um die Farbe aller Ein- bzw. Ausgangsverbindungen eines Blocks gleichzeitig zu ändern,

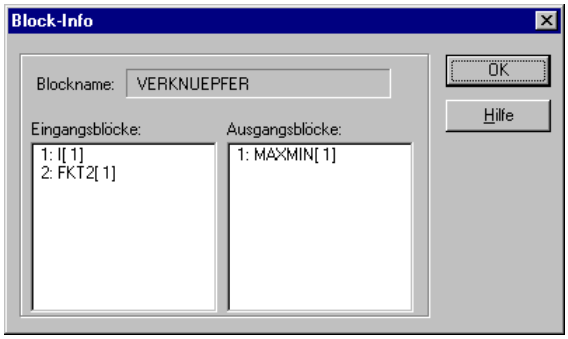
- selektieren Sie den Block und

- wählen im Hauptmenü die Option BEARBEITEN | VERBINDUNGEN | FARBE EINGANGSVERBINDUNGEN ... bzw. entsprechende.

Zur Auswahl der gewünschten Farbe erscheint jeweils der entsprechende Windows-Farbauswahl-Standarddialog.

Anzeige der Blockverbindungen

Um sich bei komplexen Systemen einen Überblick über die Verbindungen der Blöcke untereinander zu verschaffen, kann die Menüfolge BEARBEITEN | BLOCK-INFO... herangezogen werden. Der entsprechende Dialog listet dann alle mit dem selektierten Block verbundenen Blöcke - getrennt nach Ein- und Ausgängen - auf.



Block-Info-Dialog

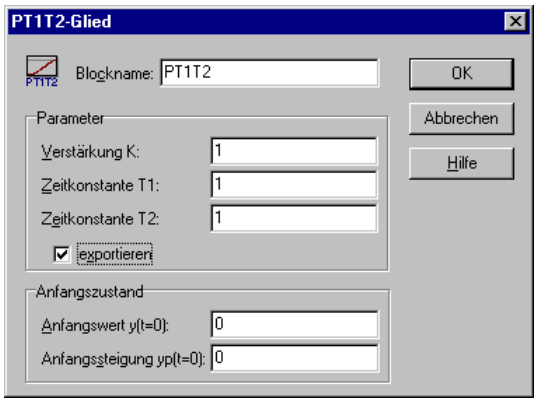
Arbeiten mit Exportparametern

Eine Vielzahl von Systemblocktypen besitzt sogenannte *Exportparameter*. Dies sind Blockparameter, die dem Anwender einen "Zugriff von außen" gestatten, der für verschiedene Zwecke genutzt werden kann:

- Aus der aktuellen Systemebene heraus können Exportparameter über einen globalen Dialog gesetzt oder auch aus einer Datei eingelesen werden (s. u.).
- Innerhalb eines Superblocks können Exportparameter benutzt werden, um diese aus dem Superblock in die darüberliegende Ebene zu exportieren und von dort zu modifizieren. Einzelheiten dazu finden Sie im Abschnitt *Arbeiten mit Superblöcken*.

- Fließkomma-Exportparameter innerhalb der aktuellen Systemebene können durch eine numerische Parameteroptimierung optimiert werden. Näheres dazu finden Sie im Abschnitt *Numerische Optimierung von Systemparametern*.

Um einen exportfähigen Blockparameter als Exportparameter aktiv werden zu lassen, muß dieser zunächst "freigeschaltet" werden. Nachfolgende Grafik zeigt den Parameterdialog eines PT_1T_2-Glieds. Bei diesem Blocktyp können die Verstärkung K sowie die Zeitkonstanten T_1 und T_2 exportiert werden. Dazu ist das Auswahlfeld *exportieren* zu aktivieren. Es sind jeweils nur alle Exportparameter eines Blocks gleichzeitig aktivierbar bzw. deaktivierbar.

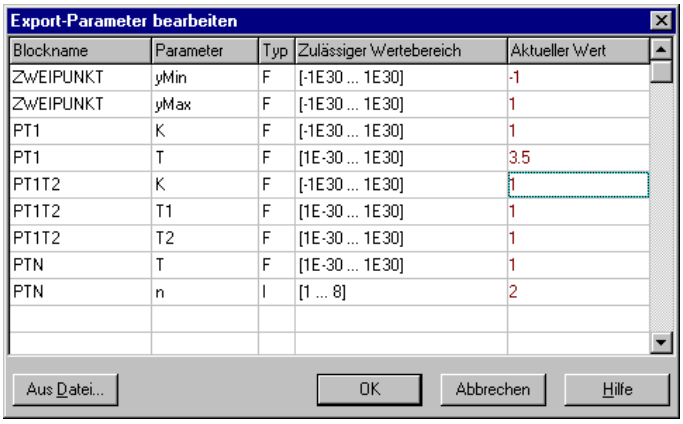


Parameterdialog eines PT1T2-Blocks mit aktivierten Exportparametern

Sollen alle verfügbaren Exportparameter der aktuellen Systemstruktur gleichzeitig aktiviert werden, so ist dies über BEARBEITEN | EXPORTPARAMETER | ALLE AKTIVIEREN möglich; über BEARBEITEN | EXPORTPARAMETER | ALLE DEAKTIVIEREN können alle Parameter wieder deaktiviert werden. Über die Menüoption BEARBEITEN | EXPORTPARAMETER | BEARBEITEN lassen sich alle aktivierten Exportparameter der aktuellen Systemstruktur modifizieren.

Die Zuordnung der Exportparameter wird dabei über den *Blocknamen* vorgenommen; alle Blöcke, die Parameter exportieren, sollten daher möglichst unterschiedliche Blocknamen aufweisen, um Fehlzuordnungen zu vermeiden. Über die Menüoption DATEI | AUF DOPPELTE BLOCKNAMEN ÜBERPRÜFEN... kann das System ggf. überprüft werden. In der Spalte *Typ* des Dialogs wird der jeweilige Parametertyp (*F* für Fließkomma, *I* für Ganzzahl und *S* für String) angezeigt. Über die Schaltfläche *Aus Datei...* können die Parameter alternativ aus einer Textdatei eingelesen werden. Diese muß in jeder Zeile einen Parameterwert

enthalten; für obiges Beispiel also in der ersten Zeile den Parameterwert für Block *Zweipunkt*, Parameter *yMin*, in der zweiten Zeile den Wert für Block *Zweipunkt*, Parameter *yMax* usw.



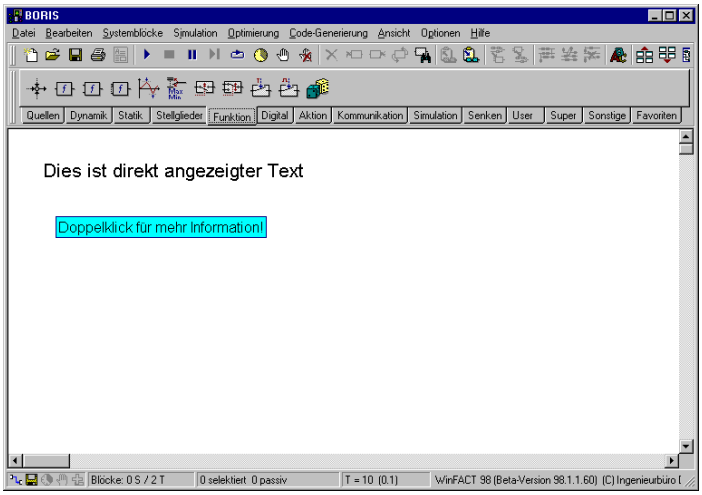
Dialog zur Bearbeitung von Exportparametern

Textblöcke und Rahmenfunktion

Arbeiten mit Textblöcken

Neben den eigentlichen Systemblöcken besteht in BORIS die Möglichkeit, beliebige Kommentartexte in das Strukturbild einzufügen. Diese Texte können in verschiedenen Farben und Schriftgrößen gewählt und ebenso wie Systemblöcke verschoben und auch wieder gelöscht werden. Wahlweise kann ein Text dabei entweder direkt auf dem Arbeitsblatt erscheinen oder aber nur ein Hinweis auf den Text selbst, der dann erst durch einen Doppelklick auf diesen Hinweis erscheint. Letztere Möglichkeit ist insbesondere dazu geeignet, längere Informationstexte in einer Systemstruktur „unterzubringen“.

Zum Einfügen eines neuen Kommentartextes dient die Menüfolge BEARBEITEN | TEXT EINFÜGEN bzw. die Schaltfläche  . Der Cursor wechselt daraufhin seine Gestalt und der voreingestellte Text *Text* kann per Drag & Drop auf dem Arbeitsblatt plaziert werden. Durch einen Doppelklick kann dieser anschließend modifiziert werden. Ein Verschieben des Textes ist wie bei Systemblökken mit festgehaltener linker Maustaste möglich.

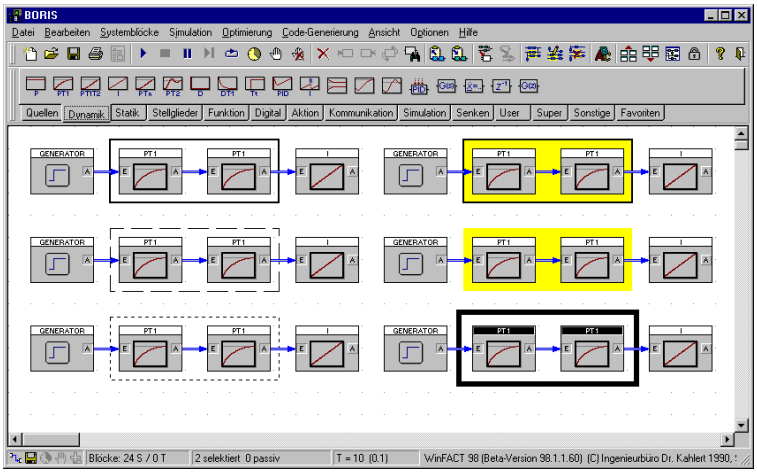


Direkt angezeigter Text (oben) und Hinweis auf versteckten Text (unten)

Dialog zur Modifikation von Textblöcken

Die Rahmenfunktion von BORIS

Ein weiteres Feature zur besseren Dokumentation von Systemstrukturen stellt die Rahmenfunktion von BORIS dar. Sie ermöglicht es, zusammengehörige Blöcke durch einen umgebenden Rahmen optisch zusammenzufassen. Auf die Simulation hat diese Funktion natürlich keinerlei Einfluß.



Verschiedene Typen von Gruppenrahmen

Um einen Rahmen einzufügen,...

- selektieren Sie zunächst die einzurahmenden Blöcke,
- wählen Sie dann die Menüfolge BEARBEITEN | GRUPPENRAHMEN | RAHMEN EINFÜGEN bzw. betätigen die Schaltfläche der System-Toolbar.

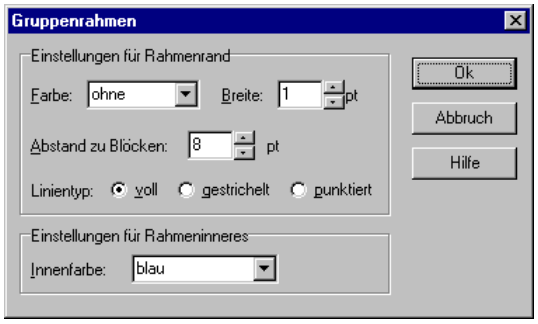
Anmerkung: Rahmen sind *statische* Gebilde, die in keinerlei direktem Bezug zu den eingerahmten Blöcken stehen. Sie können daher weder verschoben werden, noch werden sie beim Löschen von Blöcken automatisch mitgelöscht!

Erscheinungsbild des Rahmens

Der eingefügte Rahmen hat standardmäßig acht Pixel Abstand zum blockumgebenden Rechteck, einen schwarzen Rand mit Vollinie und ein Pixel Breite sowie keine Füllung. Diese Parameter lassen sich jedoch vom Anwender ändern. Dazu gehen Sie wie folgt vor:

1. Selektieren Sie einen beliebigen Block innerhalb des Rahmens.
2. Wählen Sie BEARBEITEN | GRUPPENRAHMEN | RAHMEN BEARBEITEN oder betätigen Sie die Schaltfläche der System-Toolbar.

Sie gelangen auf diese Weise in folgenden Dialog, aus dem heraus die Parameter modifizierbar sind.



Dialog zum Bearbeiten von Rahmen

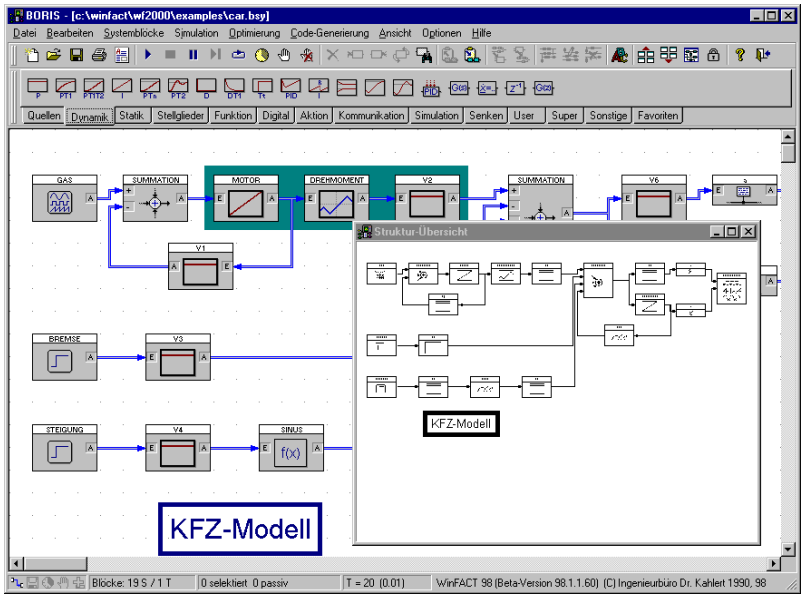
Wie Sie Rahmen wieder entfernen können

Um einen Rahmen zu löschen, gehen Sie wie folgt vor:

1. Selektieren Sie einen beliebigen Block innerhalb des Rahmens.
2. Wählen Sie BEARBEITEN | GRUPPENRAHMEN | RAHMEN LÖSCHEN oder betätigen Sie die Schaltfläche der System-Toolbar.

Struktur-Übersicht

Bei komplexeren Systemen reicht der dargestellte Bildausschnitt des Zeichenfensters in der Regel nicht zur Darstellung des Gesamtsystems aus. Um sich in solchen Fällen einen Überblick über die Gesamtstruktur zu verschaffen, besteht die Möglichkeit, ein zusätzliches Fenster mit einer maßstabsgetreuen, aber auf die Fenstergröße gezoomten Gesamtstruktur zu öffnen. Dieses Fenster erhält man über ANSICHT | STRUKTUR-ÜBERSICHT bzw. die Schaltfläche der System-Toolbar. Das Fenster kann beliebig vergrößert und verkleinert werden und wird - solange es sichtbar ist - automatisch aktualisiert. Die Systemblöcke sind innerhalb des Übersichtsfensters mit s/w-Bitmaps dargestellt.



Struktur-Übersicht für ein einfaches System

Datei-Operationen

Dateitypen

Alle BORIS-Systemdateien sind ASCII-Dateien, die gegebenenfalls mit einem normalen Texteditor betrachtet oder "nachbearbeitet" werden können (letzterer Schritt ist nur in Ausnahmefällen anzuraten!). Es ist zu unterscheiden zwischen *regulären Systemdateien* (Extension BSY) und *Superblockdateien* (Extension SBL). Letztere sind Spezialfälle von Systemdateien und enthalten zu Beginn einen Dateiheader mit speziellen Informationen über die Struktur des Superblocks. Reguläre Systemdateien können in Superblockdateien umgewandelt werden - genauso wie umgekehrt. Auf das Arbeiten mit Superblöcken und ihren Dateien wird in einem separaten Abschnitt im Detail eingegangen. In diesem Abschnitt werden lediglich reguläre Systemdateien betrachtet.

Öffnen einer Datei

Eine existierende System- oder Superblockdatei können Sie wie folgt laden:

- Wählen Sie DATEI | DATEI ÖFFNEN... bzw. betätigen Sie die Schaltfläche der System-Toolbar und geben Sie den Dateinamen an.
- Die letzten fünf bearbeiteten Dateien erscheinen am unteren Ende des DATEI-Menüs und können von dort direkt geladen werden.

Der Dateiname der aktuellen Datei - sofern sie bereits einmal abgespeichert wurde - wird in der Titelzeile des BORIS - Hauptfensters angezeigt.

Um eine Datei zu speichern...

- ... wählen Sie die Menüfolge DATEI | DATEI SPEICHERN oder betätigen die Schaltfläche der System-Toolbar. Wollen Sie die Datei unter einem neuen Namen oder einem anderen Typ (z. B. eine Superblockdatei als reguläre Systemdatei oder umgekehrt) speichern, wählen Sie stattdessen DATEI | DATEI SPEICHERN UNTER...

Anlegen einer neuen Datei

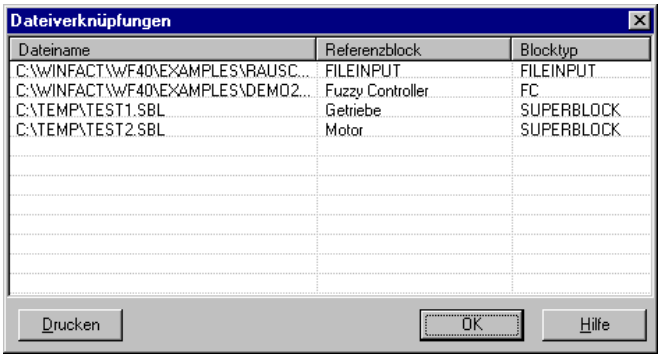
Über die Menüfolge DATEI | NEU oder die Schaltfläche wird BORIS in den Programmzustand beim Aufruf zurückversetzt. Sofern die aktuelle Datei seit der letzten Speicherung modifiziert wurde, erfolgt zuvor eine Sicherheitsabfrage.

Zusammenfügen von Dateien

Der aktuell geladenen Datei können weitere Dateien hinzugefügt werden. Zu diesem Zweck dient die Menüfolge DATEI | SYSTEMDATEI HINZUFÜGEN... mit nachfolgender Eingabe des Namens der hinzuzufügenden Systemdatei. Die zugeladene Systemstruktur wird unterhalb des aktuellen Systems angefügt. Das Hinzufügen beschränkt sich auf Blöcke, Verbindungen, Texte und Rahmen. Simulationsparameter, digitale Pegel usw. bleiben beim Hinzufügen unverändert.

Dateiverknüpfungen

Eine ganze Reihe von Systemblöcken (z. B. File-Inputs, Fuzzy Controller, Superblöcke, ...) benötigen ihrerseits Dateien für die Simulation. Soll daher beispielsweise eine Simulationsstruktur weitergegeben werden, so reicht nicht die Weitergabe der entsprechenden Systemdatei aus, sondern alle mit ihr verknüpften Dateien müssen ebenfalls mitkopiert werden. Die Menüfolge DATEI | DATEIVERKNÜPFUNGEN ... dient dabei als Hilfestellung, indem sie einen Dialog mit allen verknüpften Dateien auflistet. Angezeigt werden jeweils der Name der verknüpften Datei sowie Name und Typ des entsprechenden Systemblocks.



Anzeige von Dateiverknüpfungen

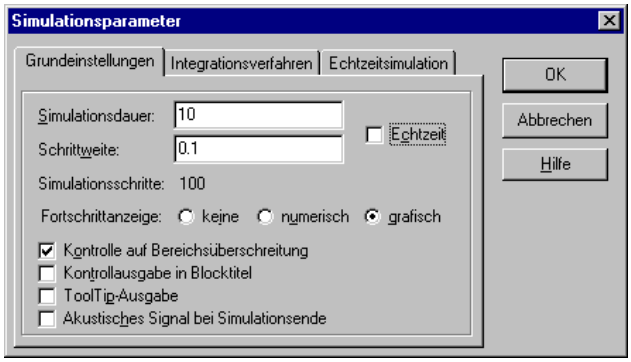
Steuerung der Simulation

Simulationsparameter

Bevor die Simulation gestartet werden kann, müssen in der Regel die Simulationsparameter gewählt werden. Der Dialog zur Wahl der Simulationsparameter

wird über SIMULATION | PARAMETER... bzw. die Schaltfläche verfügbar und ist in drei Paletten aufgeteilt.

Grundeinstellungen



Palette Grundeinstellungen

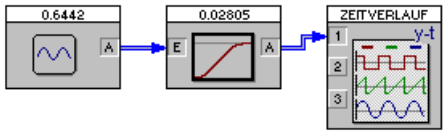
Die einzelnen Optionen haben nachfolgende Bedeutungen:

Option	Bedeutung
Simulationsdauer	Die Simulationsdauer T_{Simu} gibt an, bis zu welchem Zeitpunkt die Simulation durchgeführt wird, sofern sie nicht vorher vom Anwender abgebrochen wird.
Schrittweite	Die Simulationsschrittweite ΔT gibt die Diskretisierungsschrittweite für die Simulation an und beeinflußt damit die Genauigkeit der erhaltenen Simulationsergebnisse. Wird ΔT zu groß gewählt, entstehen Diskretisierungsfehler, die im Extremfall zu einer völligen Verfälschung der Ergebnisse durch numerische Instabilität führen können. Als Anhaltspunkt für eine geeignete Wahl gilt, daß ΔT etwa 1/10 der kleinsten im System vorkommenden Zeitkonstanten nicht überschreiten sollte (s. auch *Integrationsverfahren*).
Echtzeit	Ist dieses Optionsfeld markiert, erfolgt die Simulation in Echtzeit, d. h. die unter *Schrittweite* eingestellte Simulationsschrittweite entspricht der tatsächlichen Zeit zwischen zwei Simulationsschritten.

Fortschrittanzeige Diese Einstellung legt fest, auf welche Art der Fortlauf der Simulation angezeigt wird. In der Einstellung *grafisch* erfolgt die Anzeige in Form eines fortlaufenden Balkens in der Statuszeile, in der Einstellung *numerisch* durch Angabe der verstrichenen Zeit. Bei zeitkritischen Simulationen empfiehlt sich die Einstellung *keine*.

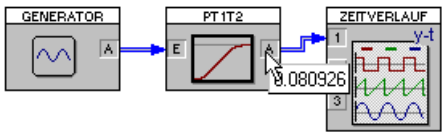
Kontrolle auf Bereichsüberschreitung Ist diese Option aktiviert, so werden alle Zustandsgrößen des Systems bei jedem Simulationsschritt überprüft. Überschreitet eine der Größen betragsmäßig den Wert 10^{20}, so wird die Simulation mit einer entsprechenden Meldung abgebrochen. Da diese Überprüfung auf Bereichsüberschreitung einen erhöhten Rechenaufwand mit sich bringt, verläuft die Simulation bei aktiver Überprüfung geringfügig langsamer.

Kontrollausgabe in Blocktitel Diese Option ist besonders nützlich bei der Suche nach Fehlern innerhalb der Systemstruktur. Ist sie aktiviert, erscheint während der Simulation im Blocktitel jedes Blocks (Ausnahme: Superblöcke) die aktuelle Ausgangsgröße des Blocks. Bei Blöcken mit mehreren Ausgängen kann allerdings nur der erste Ausgang angezeigt werden.



Kontrollausgabe in Blocktitel

ToolTip-Ausgabe Ist diese Option aktiviert, erscheint beim Überstreichen von Blockein- und -ausgängen mit der Maus ein kleines Fenster (*ToolTip*-Fenster), in dem der zugehörige Signalwert angezeigt wird.

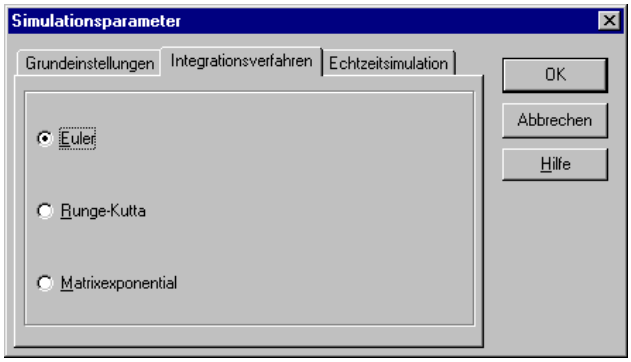


ToolTip-Ausgabe

Akustisches Signal Über diese Option kann ein Signalton bei Beendigung

bei Simulationsende bzw. Abbruch der Simulation erzeugt werden.

Integrationsverfahren



Palette Integrationsverfahren

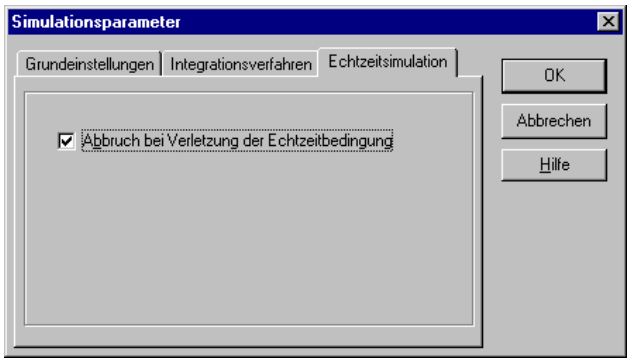
Diese Palettenseite erlaubt die Wahl des numerischen Integrationsverfahrens zur Simulation der dynamischen Systemkomponenten. In der aktuellen Version von BORIS sind das explizite Euler-Verfahren, das Runge-Kutta-Verfahren 4. Ordnung sowie das Matrizenexponentialverfahren verfügbar [3].

- Das Euler-Verfahren ist das einfachste aller Integrationsverfahren und weist damit den geringsten Rechenzeitaufwand auf, da pro Simulationsschritt nur ein Funktionswert berechnet werden muß. Es besitzt jedoch lediglich die Fehlerordnung $O(\Delta T^2)$ und eignet sich daher nur für die Simulation einfacher Systeme in Verbindung mit kleinen Simulationsschrittweiten. Speziell für stark schwingfähige Systeme oder Systeme mit stark unterschiedlichen Zeitkonstanten ("steife" Systeme) sollte das Verfahren in der Regel nicht eingesetzt werden.
- Das Runge-Kutta-Verfahren weist demgegenüber numerisch erheblich bessere Eigenschaften auf - es besitzt die lokale Fehlerordnung $O(\Delta T^5)$. Da bei diesem Verfahren jedoch für jeden Simulationsschritt vier Funktionsauswertungen erfolgen müssen, ist der Rechenzeitaufwand höher als beim Euler-Verfahren. Dies ist i. a. jedoch unerheblich. Bei sprungförmigen Änderungen von Eingangssignalen (Beispiel: Berechnung von Impulsantworten) kann es beim Einsatz dieses Verfahrens allerdings zu Ungenauigkeiten im Übergangsbereich kommen. In

diesen Fällen sollte daher besser das Euler-Verfahren in Verbindung mit einer kleineren Schrittweite eingesetzt werden.

- Das Matrizenexponentialverfahren eignet sich nur für lineare Systeme, besitzt dort jedoch eine im Prinzip unendlich hohe Fehlerordnung. Es sollte speziell dann eingesetzt werden, wenn z. B. Übertragungsfunktionen hoher Ordnung (> 3) als Systemblöcke auftreten. Wurde dieses Verfahren gewählt, so werden die nichtlinearen dynamischen Systemblöcke (nichtlineare Dgl.-Systeme) mit dem Runge-Kutta-Verfahren simuliert.

Echtzeitsimulation



Palette Echtzeitsimulation

Das Optionsfeld *Abbruch bei Verletzung der Echtzeitbedingung* bestimmt das Verhalten von BORIS bei Echtzeitsimulation für den Fall, daß die gewünschte Simulationsschrittweite aus irgendeinem Grund (z. B. wegen einer sehr komplexen Systemstruktur mit vielen rechenintensiven Blöcken) nicht eingehalten werden kann. Ist die Option aktiviert, bricht BORIS die Simulation in diesem Fall mit einer entsprechenden Warnmeldung ab.

Betriebsartensteuerung

Die Steuerung der Simulation erfolgt zweckmäßigerweise über die entsprechenden Schaltflächen der System-Toolbar, kann bei Bedarf aber auch über die jeweiligen Optionen des Untermenüs SIMULATION vonstatten gehen. Nachfolgende Tabelle gibt zunächst eine Übersicht über die entsprechenden Optionen.

Symbol	Menüoption SIMULATION |	Funktion
	START	Startet die Standard-Simulation
	STOP	Stoppt die Simulation
	PAUSE | EINZEL-SCHRITTMODUS	Aktiviert bzw. deaktiviert den Einzelschrittmodus
	EINZELSCHRITT AUSFÜHREN	Führt einen Einzelschritt aus
	START ENDLOSSIMULATION	Startet eine Endlossimulation
	BREAKPOINT SETZEN...	Setzt einen Breakpoint
	BREAKPOINT LÖSCHEN	Löscht einen Breakpoint

Starten der Standard-Simulation

In der Standard-Betriebsart wird die Simulation über die Menüfolge SIMULATION | START bzw. die System-Toolbar gestartet. BORIS überprüft nun zunächst, ob eine algebraische Schleife vorliegt oder einer der vorhandenen Systemblöcke unzulässig parametriert wurde. Gegebenenfalls wird eine entsprechende Warnung ausgegeben.

Eingänge von Systemblöcken können für die Simulation offen bleiben; sie werden in der Regel zu null bzw. in Sonderfällen auf andere Vorgabewerte gesetzt. Dadurch ist es beispielsweise bei speziellen Simulationsaufgaben (z. B. Simulation eines freien Systems mit vorgegebenen Anfangswerten) möglich, die Simulation zu starten, ohne zusätzlich einen Generator mit der Ausgangsgröße null einzusetzen.

Ist die Fehlerprüfung negativ verlaufen, so wird die Simulation gestartet. Während der Simulation wird die bereits verstrichene Simulationsdauer in der Statuszeile des Hauptfensters protokolliert, sofern diese Option aktiviert wurde.

In der Standard-Betriebsart wird die Simulation bis zu der über die Simulationsdauer vorgegebenen Zeit durchgeführt, sofern nicht zuvor ein Breakpoint erreicht wurde oder ein Benutzerabbruch erfolgte.

Endlossimulation

Im Gegensatz zur Standard-Simulation wird die vorgegebene Simulationsdauer bei einer Endlossimulation ignoriert. Die Simulation kann daher nur vom Anwender (bzw. ggf. auch über einen SIMCANCEL-Block) abgebrochen werden.

Einzelschrittsimulation

Der Anwender kann jederzeit während oder auch bereits vor der Simulation in den sogenannten *Einzelschrittmodus* wechseln. Wird der Einzelschrittmodus aktiviert, so erfolgt eine Unterbrechung der Simulation (Pause) und es kann im folgenden jeder Simulationsschritt einzeln freigegeben werden, bis der Einzelschrittmodus wieder verlassen wird. Diese Betriebsart eignet sich damit insbesondere für ein detailliertes Nachvollziehen bestimmter Simulationsbereiche. Der Einzelschrittmodus kann auch durch das Setzen von Breakpoints oder einen SIMCANCEL-Block automatisch aktiviert werden.

Abbrechen der Simulation

Die Simulation kann vom Anwender jederzeit vor ihrem regulären Ende abgebrochen werden. Dazu dient die Menüoption SIMULATION | STOP. Ist der Einzelschrittmodus aktiviert, so bleibt er dies auch nach Abbruch der Simulation!

Arbeiten mit Breakpoints

Durch das Setzen von sogenannten *Breakpoints* kann ein automatisches Umschalten in den Einzelschrittmodus zu einem festgelegten Zeitpunkt erfolgen.

Setzen eines Breakpoints (hier für t = 5)

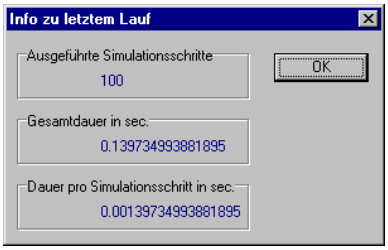
Erreicht die Simulation den vorgegebenen Zeitpunkt, erfolgt eine automatische Umschaltung in den Einzelschrittmodus. Mit Hilfe dieser Funktion läßt sich damit ein vorgegebener Zeitpunkt exakt anfahren, ohne daß die Simulation "per Hand" (und damit i. a. recht ungenau) an dieser Stelle unterbrochen wird.

Systemzustand in Anfangswerte übernehmen

Über die Option SIMULATION | SYSTEMZUSTAND IN ANFANGSWERTE ist es möglich, den aktuellen Zustand aller dynamischen Systemblöcke in deren Anfangswerte zu übernehmen, um z. B. zu einem späteren Zeitpunkt eine neue Simulation aus diesem Arbeitspunkt heraus fortzuführen. Dabei werden die aktuellen Zustandsgrößen der Blöcke in die Anfangswerte umkopiert. Diese Option ist in der aktuellen Version allerdings nur für Systemblöcke der obersten Hierarchieebene (d. h. der aktuell geladenen Systemdatei) verfügbar; die Anfangswerte von dynamischen Blöcken in Superblöcken können auf diese Weise nicht gesetzt werden! Die entsprechenden Superblöcke müssen daher ggf. vor der Simulation aufgesplittet werden.

Informationen zu letztem Lauf

Über die Option SIMULATION | INFO ZU LETZTEM LAUF... können Informationen zum vorhergehenden Simulationslauf angefordert werden. Diese Informationen sind speziell im Zusammenhang mit Echtzeitsimulationen von Interesse, da sie Hinweise auf eine kleinstzulässige Simulationsschrittweite enthalten.



Info-Dialog zu vorangegangenem Simulationslauf

Online-Parameteränderungen

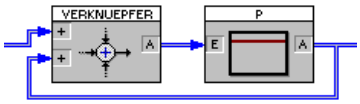
Auch Parameteränderungen von Systemblöcken sind in der Regel während der Simulation möglich (Ausnahme: Systemblöcke, die über Dateien parametriert werden, wie z. B. *File-Input*). Dazu wird der gewünschte Systemblock wie gewohnt durch einen Doppelklick angewählt und die Änderungen vorgenommen. Nach Schließen des Parameterdialogs werden alle Änderungen sofort wirksam und die Simulation wird fortgeführt. Änderungen der Systemstruktur (z. B. Löschen von Blöcken oder Verbindungen) sind während der Simulation nicht möglich.

Was tun bei Algebraischen Schleifen?

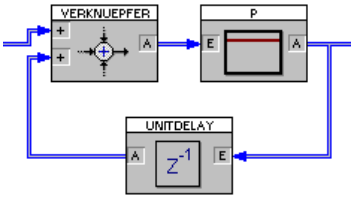
Nachdem Sie einen Simulationslauf gestartet haben, sortiert BORIS alle Systemblöcke intern zunächst in eine für die Simulation geeignete Reihenfolge. Dies ist erforderlich, um unabhängig von der Reihenfolge, in der die Blöcke eingefügt wurden, immer reproduzierbare, korrekte Ergebnisse zu erhalten. Dieser Sortiervorgang schlägt jedoch fehl, wenn sich innerhalb Ihrer Simulationsstruktur eine sogenannte *Algebraische Schleife* befindet. Dies ist eine geschlossene Struktur, also eine Struktur mit Rückführung (Schleife), bei der sich innerhalb dieser Schleife nur Systemblöcke befinden, die nicht verzögernd wirken, d. h. bei denen sich eine Änderung am Eingang noch im selben Simulationsschritt am Ausgang bemerkbar macht. Damit BORIS die Blöcke sortieren kann, muß sich innerhalb einer Schleife also immer mindestens ein Block mit Verzögerung befinden. Dazu zählen folgende Systemblocktypen:

- PT_1, PT_2, PT_1T_2, PT_n - Glieder
- Integrierer und Totzeitglieder
- Allpaßglieder
- Übertragungsfunktionen mit Zählergrad < Nennergrad
- Einheitsverzögerungen

Tritt innerhalb Ihrer Struktur eine algebraische Schleife auf, macht BORIS Sie durch eine entsprechende Meldung darauf aufmerksam. Sie sollten in diesem Fall Ihr Modell zunächst überprüfen, da algebraische Schleifen in der Regel auf Modellierungsfehler hinweisen (in der Realität gibt es keine algebraischen Schleifen!). Wollen Sie die Struktur dennoch simulieren, können Sie zur Abhilfe einen der oben aufgeführten Verzögerungsblöcke in die Schleife einfügen; am besten eignet sich dazu eine Einheitsverzögerung, bei der das Eingangssignal genau um einen Takt verzögert wird oder ein PT_1-Glied mit einer kleinen Zeitkonstanten. Nachfolgende Grafiken verdeutlichen das Prinzip.



Beispiel für eine Algebraische Schleife...



...und ihre Aufhebung durch Einfügen einer Einheitsverzögerung

Was Sie sonst noch wissen sollten

Verwaltung von Alarmen/Meldungen

BORIS besitzt eine leistungsfähige Meldungs- bzw. Alarmverwaltung, mit der Sie während der Simulation anfallende Meldungen/Alarme zentral erfassen und anzeigen können. Dazu dient ein Meldungsfenster, das über die Option ANSICHT | MELDUNGEN/ALARME jederzeit angezeigt werden kann. Zur Generierung von Meldungen fügen Sie einfach an entsprechenden Stellen einen Meldungs-Block ein. Jede Meldung wird im Klartext mit dem Namen des entsprechenden Meldungsblocks sowie Simulationszeit oder Echtzeit/Datum beim Auftreten der Meldung angezeigt und kann darüber hinaus auch akustisch in Form einer WAV-Datei unterlegt werden. Das Meldungsfenster bietet neben der reinen Anzeigefunktion folgende Optionen:

- Durch Aktivierung von *Online-Speicherung in BORIS.MSG* werden Meldungen automatisch in der Datei BORIS.MSG gespeichert.
- Vor dem Beginn einer neuen Simulation können die angezeigten Meldungen automatisch gelöscht werden (Option *Löschen vor jeder Simulation*).
- Das Meldungsfenster kann zu Beginn der Simulation automatisch angezeigt werden (Option *Fenster autom. anzeigen*; Anzeige erfolgt nur, wenn System mindestens einen Meldungs-Block enthält!).

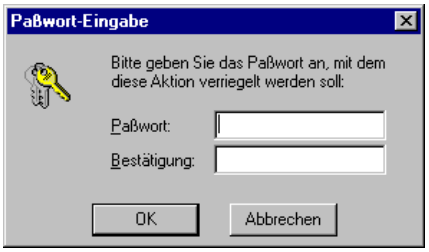
- Beim Eintreffen einer Meldung kann ein Signalton erzeugt werden (Option *Signalton bei Meldung*).
- Alle angezeigten Meldungen können manuell gelöscht und in einer beliebigen ASCII-Datei gespeichert werden (Schalter *Meldungen löschen* bzw. *Meldungen speichern.....*).

Verwaltungsfenster für Meldungen/Alarme

Die Priorität der Meldung wird durch die Farbe des Ausrufezeichens vor dem Meldungstext gekennzeichnet. Dabei weist gelb Meldungen mit niedriger Priorität, grün Meldungen mit mittlerer Priorität und rot Meldungen mit hoher Priorität aus. Weitere Einzelheiten entnehmen Sie bitte der Beschreibung des Meldungsblocks im Abschnitt *Die BORIS-Systemblock-Bibliothek*.

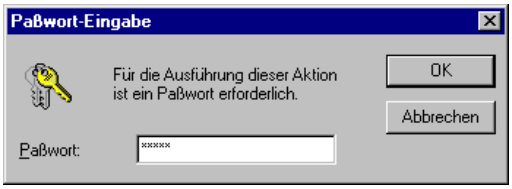
Verriegeln des Hauptfensters

Um z. B. während länger andauernder Simulationen oder Meßwerterfassungen unbefugte Zugriffe zu verhindern, kann das BORIS-Hauptfenster verriegelt werden. Dies wird über die Menüfolge ANSICHT | VERRIEGELN bzw. die Schaltfläche bewerkstelligt. BORIS fragt anschließend nach einem Paßwort, welches später zur Wiederherstellung des Hauptfensters dient.



Eingabe des Paßworts zum Verriegeln des Hauptfensters

Das Paßwort muß zur Sicherung vor Fehleingaben in einem zweiten Eingabefeld bestätigt werden. Anschließend wird das Hauptfenster zum Symbol verkleinert und kann danach nur durch erneute Eingabe des Paßworts wieder regeneriert werden.



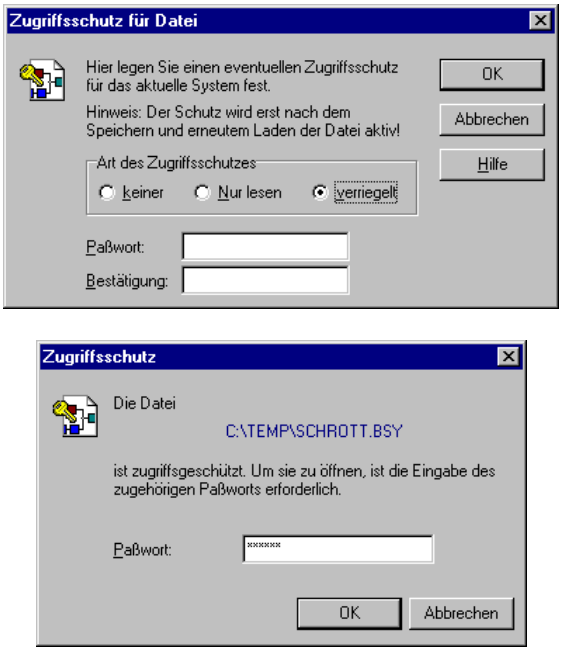
Eingabe des Paßworts zum Entriegeln des Hauptfensters

Zugriffsschutz für Systemdateien

BORIS-System- und Superblockdateien können mit einem Zugriffsschutz versehen werden, der ein unbefugtes Modifizieren oder Öffnen der Datei verhindert. Dazu dient die Menüoption DATEI | ZUGRIFFSSCHUTZ... Es sind drei Stufen des Zugriffsschutzes verfügbar:

keiner	Keine Einschränkungen für Dateizugriff
nur lesen	Eine Modifikation des Systems ist nur nach Eingabe des korrekten Paßworts möglich. Wird beim Öffnen der Datei kein Paßwort bzw. ein falsches Paßwort angegeben, kann das System nur gelesen und simuliert werden.
verriegelt	Ein Öffnen des Systems ist nur nach Eingabe des korrekten Paßworts möglich.

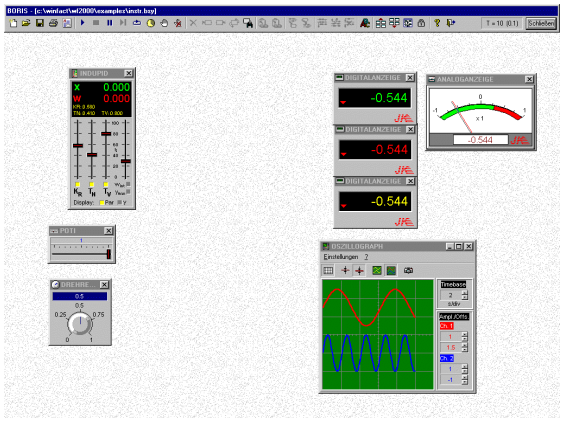
Das eingegebene Paßwort muß zur Sicherung vor Fehleingaben im unteren Eingabefeld noch einmal bestätigt werden. Beachten Sie bitte, daß ein eventueller Zugriffsschutz erst nach dem Speichern beim folgenden Ladeversuch aktiviert wird.



Festlegen des Zugriffsschutzes für System- bzw. Superblockdateien (oben) und Paßwort-Abfrage beim Öffnen einer Datei mit Zugriffsschutz (unten)

Wechsel des Anzeigemodus

Für bestimmte Anwendungsfälle von BORIS (z. B. Meßdatenerfassungen) kann es wünschenswert sein, das eigentliche Hauptfenster mit der Systemstruktur ausblenden zu können und nur die Aktions- bzw. Anzeigeblöcke auf dem Bildschirm erscheinen zu lassen. BORIS bietet zu diesem Zweck einen alternativen Anzeigemodus an, der über ANSICHT | ANZEIGEMODUS WECHSELN aktiviert werden kann. Das BORIS-Hauptfenster wird dabei bis zu einer modifizierten System-Toolbar verkleinert und mit einem neutralen Hintergrund versehen. Über die *Schließen*-Schaltfläche kann zurück in den Standardmodus gewechselt werden.

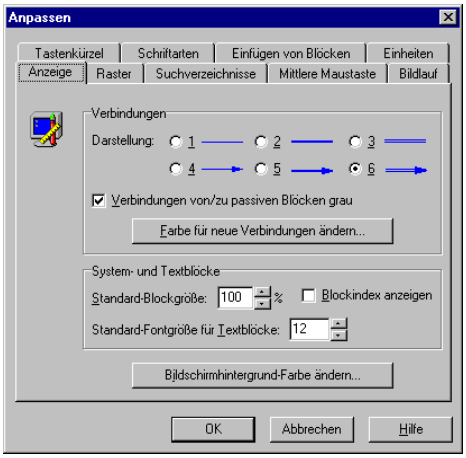


Alternativer Anzeigemodus

Benutzerdefinierte Einstellungen

BORIS erlaubt Ihnen über die Menüoption OPTIONEN | ANPASSEN... die Vorgabe einer Reihe von benutzerdefinierten Einstellungen.

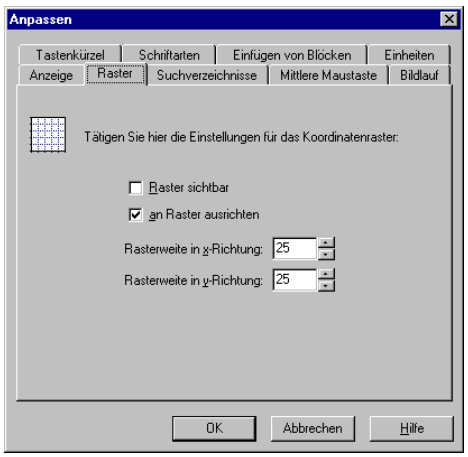
Anzeige-Optionen



Über die Palette *Anzeige* sind einige Optionen erreichbar, die insbesondere die Bildschirmanzeige betreffen:

Option	Bedeutung
Darstellung	Darstellungsart von Verbindungen
Verbindungen von/zu passiven Blöcken grau	Darstellungsart von Verbindungen von/zu passiv gesetzten Blöcken
Farbe für neue Verbindungen ändern	Ermöglicht die Wahl einer neuen voreingestellten Farbe für zukünftig eingefügte Blöcke. Diese Option beeinflußt nicht bereits vorhandene Verbindungen!
Standard-Blockgröße	Ermöglicht die Wahl einer neuen voreingestellten Größe für zukünftig eingefügte Blöcke. Diese Option beeinflußt nicht bereits vorhandene Böcke!
Blockindex anzeigen	Ist diese Option aktiviert, wird im Blocktitel eines Blocks neben dem Blocknamen auch der Blockindex angezeigt.
Standard-Fontgröße für Textblöcke	Ermöglicht die Wahl einer neuen voreingestellten Schriftgröße für zukünftig eingefügte Textblöcke. Diese Option beeinflußt nicht bereits vorhandene Textblöcke!
Bildschirmhinter-grund-Farbe ändern	Über diese Schaltfläche kann die Farbe für den Hintergrund der BORIS-Zeichenfläche modifiziert werden.

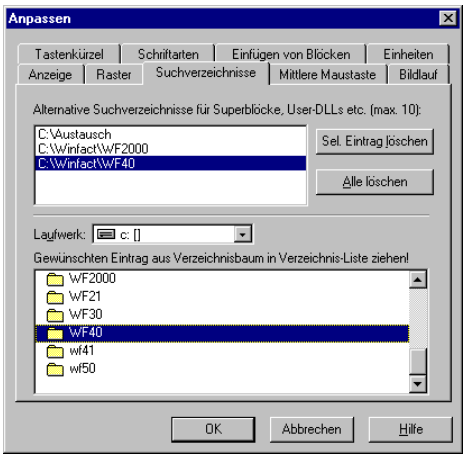
Raster



Die Optionen dieser Palette haben nachfolgende Bedeutung.

Option	Bedeutung
Raster sichtbar	Schaltet die Anzeige des Bildschirmrasters zu bzw. ab
An Raster ausrichten	Aktiviert bzw. deaktiviert die Ausrichtung der Systemblöcke am Raster. Die Ausrichtung ist unabhängig davon, ob das Raster sichtbar ist.
Rasterweite in x-Richtung	Legt den horizontalen Abstand zwischen zwei Rasterpunkten in Pixeln fest
Rasterweite in y-Richtung	Legt den vertikalen Abstand zwischen zwei Rasterpunkten in Pixeln fest

Suchverzeichnisse



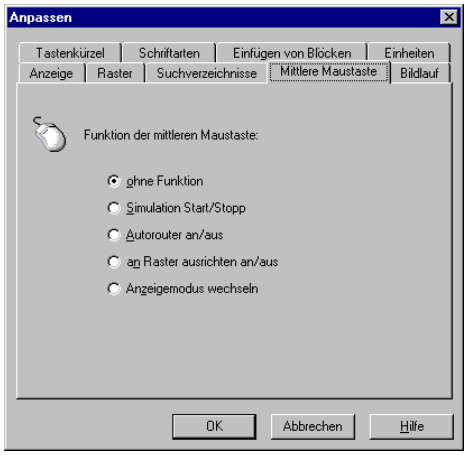
BORIS erlaubt Ihnen über die Menüoption OPTIONEN | ANPASSEN..., Palette *Suchverzeichnisse*, die Vorgabe von bis zu zehn Suchpfaden. In diesen Pfaden werden (in der Reihenfolge des Eintrags) alle dateibasierten Blöcke (also z. B. Superblöcke oder User-Blöcke) gesucht, sofern die vom Anwender vorgegebenen Namen keinen oder aber einen nicht vorhandenen Pfad enthalten. Einen neuen Eintrag fügen Sie einfach ein, indem Sie das gewünschte Verzeichnis aus dem Verzeichnisbaum im unteren Bereich des Dialogs per Drag & Drop in die obere Liste ziehen und dort fallenlassen. Über die Schaltfläche *Sel. Eintrag*

löschen kann der gerade selektierte Eintrag gelöscht werden, über *Alle löschen* werden alle Einträge gelöscht.

Die Definition von Suchverzeichnissen ist insbesondere dann angebracht, wenn komplexe Systemstrukturen mit vielen Superblöcken, User-DLLs usw. auf einen anderen Rechner mit anderer Verzeichnisstruktur portiert werden sollen. Es ist dann nicht erforderlich, alle Pfadangaben in den entsprechenden Systemblöcken mühsam per Hand zu ändern, sondern die Definition eines oder einiger weniger Suchverzeichnisse reicht aus.

Beispiel: Es sei auf Rechner A eine BSY-Datei im Verzeichnis C:\WINFACT erstellt worden, die diverse Superblöcke im Unterverzeichnis C:\WINFACT\SB enthalte. Diese Simulationsstruktur soll nun auf Rechner B portiert werden, auf dem BSY-Dateien gewöhnlich im Verzeichnis C:\WF und Superblockdateien im Unterverzeichnis C:\WF\SUPER liegen. Dann reicht es aus, diese beiden Pfadnamen auf Rechner B als Suchverzeichnisse anzugeben und die Simulation kann unmittelbar gestartet werden. Beim nächsten Abspeichern des Systems auf Rechner B werden alle Dateinamen dann außerdem automatisch angepaßt.

Konfigurierung der mittleren Maustaste

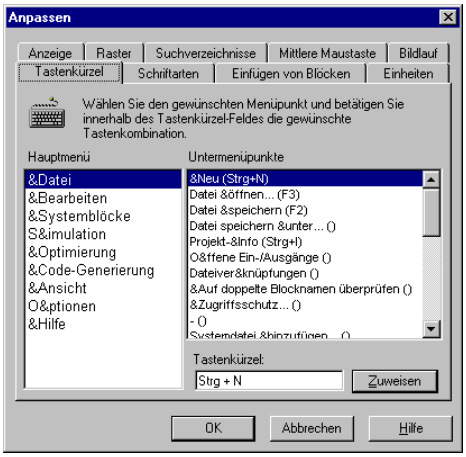


Diese Palette erlaubt Anwendern mit einer Drei-Tasten-Maus die Konfigurierung der mittleren Maustaste auf eine bestimmte Funktion.

Bildlauf

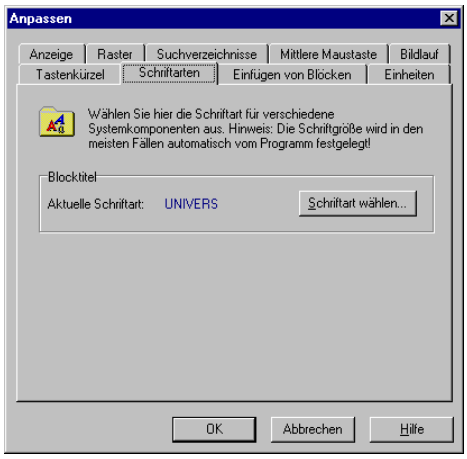
Diese Palette ermöglicht die Anpassung der Bildlaufgeschwindigkeit (Scrollgeschwindigkeit) beim Verschieben von Blöcken etc. an die benutzte Rechnerkonfiguration, insbesondere an die verwendete Grafikkarte. Wird beim Verschiebevorgang zusätzlich die <Shift>-Taste gedrückt, erfolgt der Bildlauf außerdem mit doppelter Geschwindigkeit.

Tastenkürzel



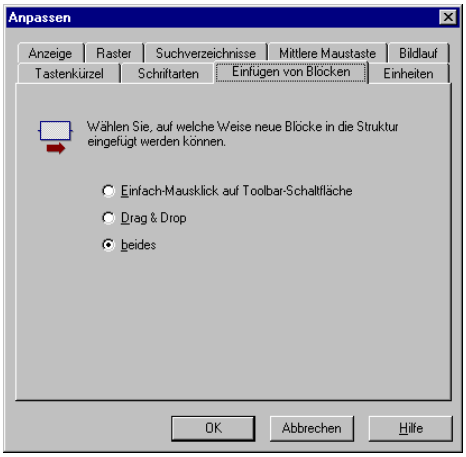
Über diese Palette lassen sich die Tastenkürzel (Shortcuts) für sämtliche Befehle des BORIS-Hauptmenüs an die eigenen Gewohnheiten anpassen. Dazu wird zunächst der gewünschte Menüpunkt über die beiden Listen *Hauptmenü* und *Untermenüpunkte* ausgewählt. Im Eingabefeld *Tastenkürzel* kann dann durch die direkte Einagbe der gewünschten Taste oder Tastenkombination (z. B. <Strg> <A>) und anschließende Betätigung der *Zuweisen*-Schaltfläche die Zuordnung getroffen werden.

Schriftarten



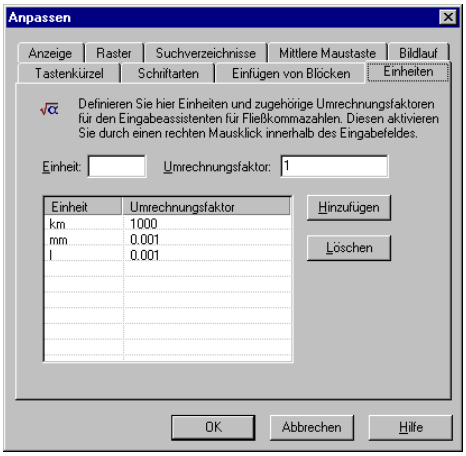
Je nach Rechner-Betriebssystem und installierten Schriftarten können die Blocktitel, die in der Schriftgröße 6 pt angezeigt werden, gelegentlich schlecht lesbar erscheinen. Daher kann über diese Palette die am besten geeignete Schriftart für die Blocktitel ausgewählt werden.

Einfügen von Blöcken

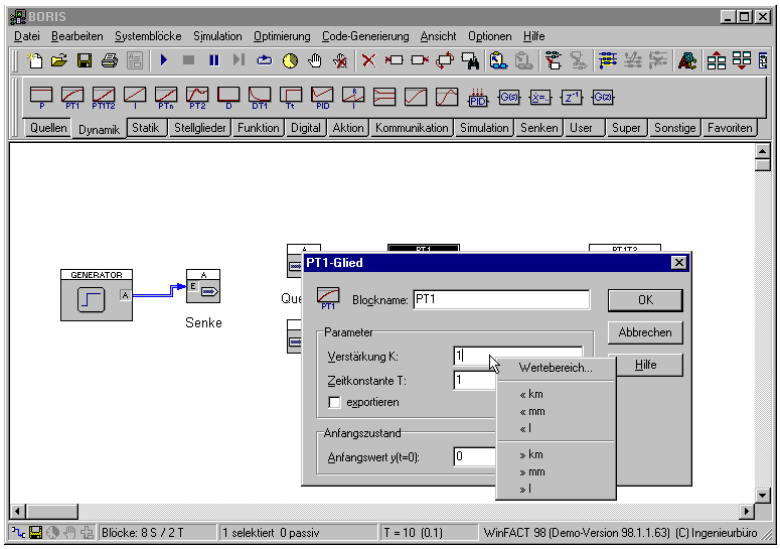


BORIS erlaubt das Einfügen neuer Blöcke sowohl über einen Einfachklick auf die entsprechende Schaltfläche der Systemblock-Toolbar (wobei der eingefügte Block dann automatisch plaziert wird), als auch per Drag & Drop. Beide Einfügeoptionen können über diese Palette getrennt voneinander aktiviert bzw. deaktiviert werden.

Einheiten



Um die Umrechnung von Größen, die in unterschiedlichen Einheiten vorliegen (z. B. m^3/s und l/h), besitzt BORIS einen integrierten Eingabeassistenten, der automatisch aktiviert wird, wenn innerhalb des Eingabefeldes für eine Fließkommazahl die rechte Maustaste gedrückt wird. In dem daraufhin erscheinenden Popup-Menü kann auf die eingegebene Zahl direkt ein einheitenspezifischer Umrechnungsfaktor oder sein Kehrwert angewendet werden. Über die Palette *Einheiten* können diese Einheiten zusammen mit ihren Umrechnungsfaktoren konfiguriert werden. Dazu wird im Eingabefeld *Einheit* die Einheit und im Feld *Umrechnungsfaktor* der zugehörige Umrechnungsfaktor angegeben. Über die Schaltfläche *Hinzufügen* wird die Einheit in die Liste aufgenommen. Der zugehörige Kehrwert wird von BORIS automatisch erzeugt.



Aufruf des Eingabeassistenten für Fließkommazahlen

Konfigurierung der Systemblock-Toolbar

Die Systemblock-Toolbar kann vom Anwender an seine Bedürfnisse angepaßt werden. Der zugehörige Dialog, der in drei Paletten aufgeteilt ist, wird über die Menüoption OPTIONEN | TOOLBARS | SYSTEMBLOCKTOOLBAR KONFIGURIEREN... verfügbar.

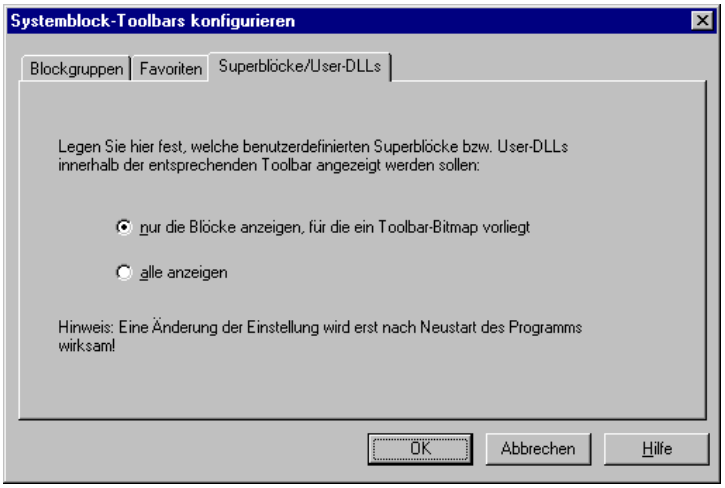
Anzuzeigende Blockgruppen

Die Systemblock-Toolbar ist aufgeteilt in mehrere Paletten, die jeweils zusammengehörige Blocktypen umfassen. Über die Palette *Blockgruppen* lassen sich einzelne Paletten der Systemblock-Toolbar bei Bedarf ausschalten. Standardmäßig sind sämtliche Paletten sichtbar.

Palette Favoriten

Die *Favoriten*-Palette der Systemblock-Toolbar kann vom Anwender über diese Dialogpalette frei konfiguriert werden. Dazu werden die gewünschten Blocktypen aus dem unteren Listenfenster einfach per Drag & Drop auf die darüberliegende Toolbar gezogen. Standardmäßig ist die *Favoriten*-Palette leer.

Superblöcke und User-DLLs



Die Paletten *User* bzw. *Super* der Systemblock-Toolbar enthalten neben jeweils „leeren“ Blöcken des entsprechenden Typs auch alle benutzerdefinierten Superblöcke bzw. User-DLLs, die BORIS im WinFACT 98-Programmverzeichnis oder in einem der Suchverzeichnisse findet. Über diese Dialogpalette wird festgelegt, ob dabei sämtliche benutzerdefinierten Blöcke aufgenommen werden, oder nur diejenigen, für die ein benutzerdefiniertes Toolbar-Bitmap gefunden wurde. Beachten Sie dabei, daß eine Änderung dieser Einstellung erst beim nächsten Aufruf von BORIS aktiv wird.

Starten von BORIS mit Aufrufparametern

BORIS kann optional mit Aufrufparametern gestartet werden, die ein automatisches Einlesen einer Datei und bei Bedarf auch einen automatischen Simulationsstart bewirken:

BORIS *Dateiname* startet BORIS mit der Datei *Dateiname*.

BORIS *Dateiname* /S	startet BORIS, lädt automatisch die Datei *Dateiname* und startet die Simulation. Der Schalter */S* kann wahlweise auch vor dem Dateinamen stehen.
BORIS *Dateiname* /E	startet BORIS, lädt automatisch die Datei *Dateiname* und startet eine Endlossimulation. Der Schalter */E* kann wahlweise auch vor dem Dateinamen stehen.

Die BORIS-Systemblock-Bibliothek

Arten von Systemblöcken

In den folgenden Abschnitten werden alle Module der BORIS-Systemblockbibliothek beschrieben. Die Systemblocktypen sind aufgeteilt in folgende Typklassen, die jeweils auf einer eigenen Palette der Systemblock-Toolbar zusammengefaßt sind:

Quellen

In diese Gruppe gehören diejenigen Systemblocktypen, die Eingangssignale für das System liefern (z. B. Generator, Datei-Eingabe).

Dynamik

Hierunter fallen alle linearen und nichtlinearen dynamischen Systeme, beginnend beim einfachen P-Glied bis hin zu frei parametrierbaren Übertragungsfunktionen und Dgl.-Systemen.

Statik

Dies sind im wesentlichen nichtlineare Kennlinien- bzw. Kennfeldglieder (Beispiele: Begrenzer, Tote Zone).

Stellglieder

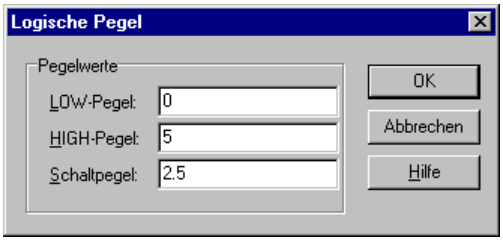
Diese Klasse enthält Blöcke, die das Verhalten realer industrieller Stellglieder nachbilden.

Funktion

Als Funktionsblöcke werden diejenigen Blocktypen bezeichnet, die nicht als Übertragungssysteme im herkömmlichen Sinne anzusehen sind. Hierzu gehören insbesondere Summierer und andere Verknüpfer mehrerer Eingangsgrößen.

Digital

Diese Klasse enthält solche Blöcke, deren Ausgangsgröße nur die binären Zustände high (logisch 1) bzw. low (logisch 0) annehmen kann. Die entsprechenden Pegel können über SYSTEMBLÖCKE | DIGITALBLÖCKE | PEGEL vorgegeben werden.



Vorgabe der Logik-Pegel für Digitalbausteine

Der Wert für den *Schaltpegel* gibt dabei den Schwellwert an, ab dem ein digitaler Eingangswert als logisch 1 angesehen wird. Voreingestellt sind die Werte 0 für low, 5 für high und 2.5 für den Umschaltwert (entsprechend TTL).

Aktion

Dies sind Blöcke, die dem Benutzer einen interaktiven Eingriff gestatten (z. B. Schalter).

Kommunikation

Diese Gruppe umfaßt Blöcke zur Kommunikation über DDE oder das TCP/IP-Protokoll.

Simulation

Hierunter fallen Blöcke, die die Simulationssteuerung selbst betreffen.

Senken

Diese Typklasse umfaßt all jene Blöcke, die lediglich einen bzw. mehrere Eingänge besitzen. Dies sind in der Regel Blocktypen zur Visualisierung oder Weiterverarbeitung von Simulationsergebnissen (Beispiel: Oszillograph, File-Output). Erstere weisen neben dem Systemblock im Zeichenfenster zusätzlich noch das eigentliche *Anzeigefenster* auf, innerhalb dessen die Simulationsergebnisse dargestellt werden.[1] Beim Einfügen eines solchen Blocks wird dieses Anzeigefenster zunächst in Symboldarstellung mit einem typspezifischen Icon dargestellt. Zur Simulation können alle Anzeigefenster über ANSICHT | ALLE ANZEIGEFENSTER ZEIGEN in Normalgröße dargestellt werden. Entsprechend können sie über ANSICHT | ALLE ANZEIGEFENSTER VERBERGEN jederzeit wieder in die Symboldarstellung zurückversetzt werden. Das Anzeigefenster weist jeweils den gleichen Titel auf wie der Systemblock selbst. Die Parametrierung von Ausgangsblöcken kann wahlweise durch einen Doppelklick auf den Systemblock oder innerhalb des Anzeigefensters erfolgen. Bei einigen der Ausgangsblöcke sind die Anzeigefenster zudem in ihrer Größe veränderbar (z. B. Oszillograph).

Sonstige

Diese Klasse enthält alle Blöcke, die nicht eindeutig einer der vorgenannten Gruppen zugeordnet werden können.

Hinweis: Die in den folgenden Abschnitten mit einem Sternchen (*) versehenen Blocktypen sind nur bei Erwerb der entsprechenden BORIS-Zusatzmodule bzw. Hardwarekomponenten und -treiber verfügbar!

Eingangsblöcke (Quellen)

Generator

Typname: GENERATOR

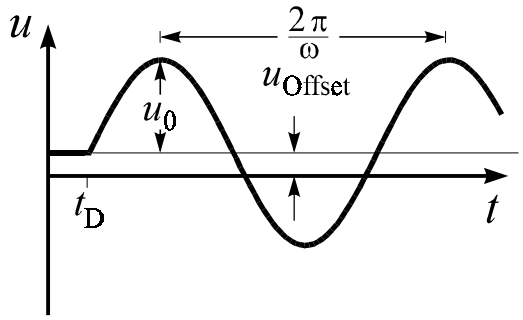
[1] Auch andere Blocktypen (z. B. Fuzzy Controller) können ein Anzeigefenster besitzen.

Funktion: Dient zur Erzeugung von Testsignalen verschiedenster Art zur Anregung von Systemen. Der Generator weist verschiedene Betriebsarten auf, die in der Systemstruktur jeweils durch unterschiedliche Block-Bitmaps gekennzeichnet werden:

Sinusgenerator

In diesem Fall erzeugt der Generator ein sinusförmiges Ausgangssignal der Form

$$u(t) = \begin{cases} u_{\text{Offset}} & \textit{für } t < t_{\text{D}} \\ u_0 \sin(\omega(t - t_{\text{D}}) + \varphi) + u_{\text{Offset}} & \textit{sonst} \end{cases}$$

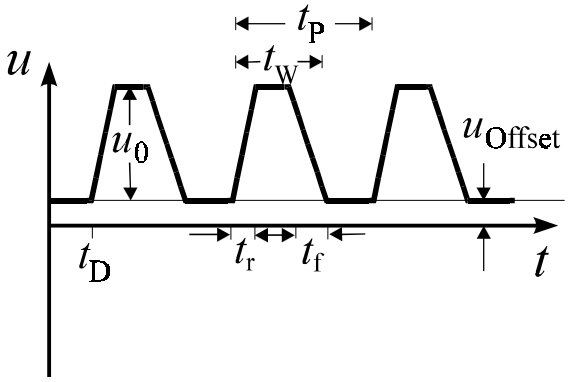


Parameter:

u_0 :	Amplitude
u_{Offset} :	Offset
t_{D} :	Verzugszeit
ω :	Kreisfrequenz
φ :	Phase (in Grad)

Pulsgenerator

In diesem Fall erhält man je nach Parameterwahl Rechteckimpulse, Dreieckimpulse oder auch Sägezahnimpulse:



Parameter:
- t_r : Anstiegszeit
- t_f : Abstiegszeit
- t_w : Pulsweite
- t_p : Periodendauer

Restliche Parameter wie bei Sinusgenerator

Durch spezielle Wahl der Parameter lassen sich auch Einzelimpulse erzeugen. Wählt man beispielsweise Pulsweite und Periodendauer größer als die Simulationsdauer und An- bzw. Abstiegszeit zu null, so erhält man eine Sprungfunktion (dies ist die Voreinstellung!).

Rauschgenerator

In diesem Fall erzeugt der Generator zu jedem Simulationszeitpunkt eine gleichverteilte Zufallszahl aus dem Intervall $[\, u_{Offset}\, ,\, u_{Offset} + u_0\,]$.

Programmierbarer Funktionsgenerator

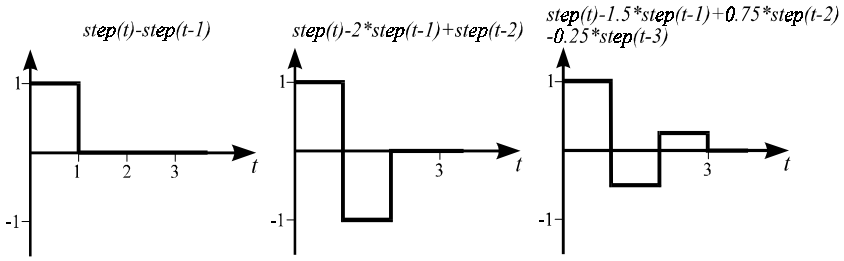
Für spezielle Anwendungen kann der Generator vom Anwender über eine nahezu beliebige Funktion programmiert werden, die über einen Funktionsparser interpretiert wird. Diese Betriebsart ist allerdings relativ rechenzeitaufwendig. Der Parser erlaubt die Interpretation folgender Symbole bzw. Funktionen:

Syntax	**Funktion**
t	Unabhängige Variable (Zeit)
+	Plus (Addition)
-	Minus und monadisches Minus

*	Multiplikation
/	Division
^	Potenzoperator
( )	Klammern (max. Verschachtelungstiefe 20)
ln	natürlicher Logarithmus
log	Zehnerlogarithmus
sqr	Quadrat
sqrt	Wurzel
exp	Exponentialfunktion
sin	Sinus
cos	Cosinus
tan	Tangens
asin	Arcussinus
acos	Arcuscosinus
atan	Arcustangens
sinh	Sinus hyperbolicus
cosh	Cosinus hyperbolicus
tanh	Tangens hyperbolicus
abs	Absolutwert
int	ganzzahliger Anteil
frac	gebrochener Anteil
round	rundet auf ganze Zahl
sign	Signumfunktion
step	Sprungfunktion (Heavisidefunktion)

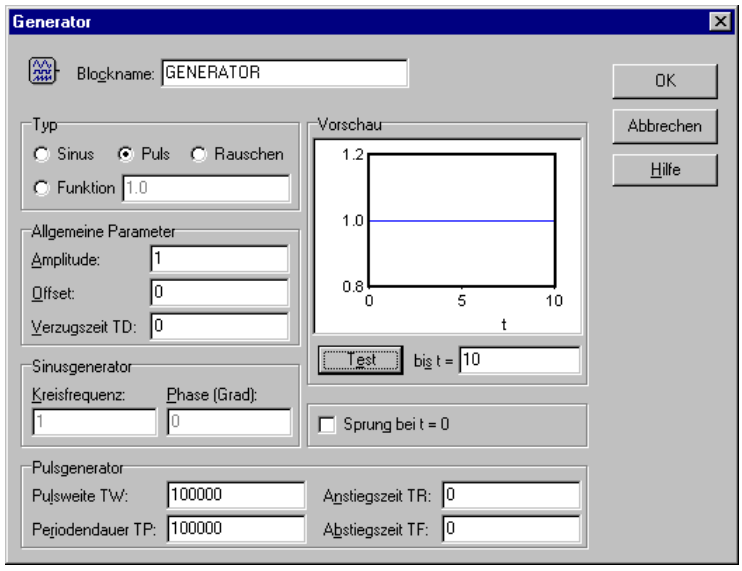
Der Funktionsstring darf maximal 255 Zeichen aufweisen. Groß- und Kleinschreibung werden nicht unterschieden.

Über den Funktionsgenerator lassen sich in Verbindung mit der Sprungfunktion z. B. auf einfache Weise nahezu beliebige Impulsformen realisieren. Die nachfolgenden Kurven zeigen einige Beispiele jeweils zusammen mit dem entsprechenden Funktionsaufruf.



Erzeugung diverser Impulsformen über die step-Funktion

Parameter-dialog:



Die Schaltergruppe *Typ* dient zur Wahl der Betriebsart. Die aktuellen Einstellungen können über die Schaltfläche *Test* jederzeit ausgetestet werden.

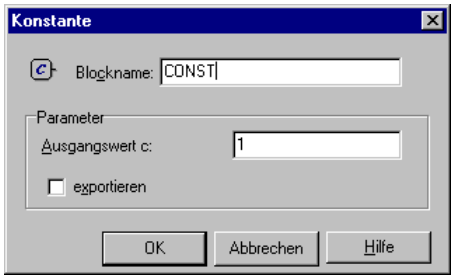
Ist das Schaltfeld *Sprung bei t=0* markiert, so wird die Ausgangsgröße des Generators unabhängig von den gewählten Einstellungen zum Simulationsbeginn auf null gesetzt. Die eigentliche Ausgangsgröße wird in diesem Fall erst beim ersten Simulationsschritt, d. h. nach einer Simulationsschrittweite ΔT aufgeschaltet.

Konstante

Typname: CONST

Funktion: Gibt einen konstanten Wert aus.

Parameter-dialog:

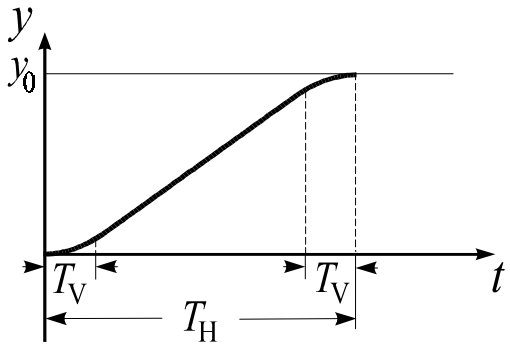


Ausgangswert gibt den ausgegebenen Wert an.

Fahrkurve

Typname: FAHRKURVE

Funktion: Der Eingangsblock erzeugt ein gleichmäßig von null auf den Endwert ansteigendes Signal, wie es beispielsweise zum Hochfahren von Motoren benutzt wird. Nachfolgende Grafik zeigt den prinzipiellen Verlauf des Signals.



Es gilt für den Verlauf von $y(t)$:

$$y(t) = \frac{y_0 t^2}{2(T_H - T_V)T_V}\Bigg|_{t=0}^{t=T_V} + \frac{y_0}{T_H - T_V}(t - T_V)\Bigg|_{t=T_V}^{t=T_H - T_V} + \ldots$$
$$\ldots + \frac{y_0 T_V}{2(T_H - T_V)}\left(1 - \frac{(t - T_H)^2}{T_V^2}\right)\Bigg|_{t=T_H - T_V}^{t=T_H}$$

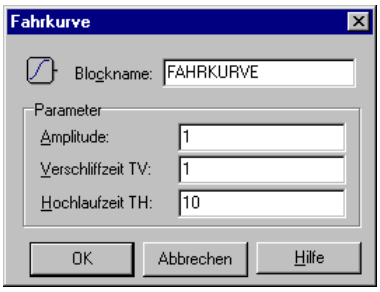
Darin ist:

y_0: Amplitude des Signals

T_V: Verschliffzeit

T_H: Hochlaufzeit

Parameter-dialog:



Parameter-grenzen: $T_H \geq 2T_V$

Steuerbarer Sinusgenerator (VCO)

Typname: VCO

Funktion: Dieser Block stellt einen Sinusgenerator dar, dessen Frequenz über die Eingangsgröße variiert werden kann. Die Variation kann linear oder exponentiell erfolgen.

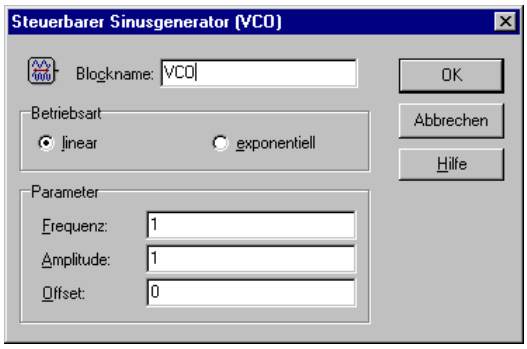
Die Grundwerte entsprechen den Einstellungen des Blocktyps *Generator* in der Betriebsart als Sinusgenerator. Ist ω_0 die Grundfrequenz des VCO, so ergibt

sich die aktuelle Frequenz ω in Abhängigkeit von der Eingangsgröße x wie folgt:

Betriebsart *linear*: $\omega = x\omega_0$

Betriebsart *exponentiell*: $\omega = 10^x \omega_0$

Parameter-dialog:



Simulationszeit

Typname: SIMTIME

Funktion: Dieser Block gibt die verstrichene Simulationszeit aus.

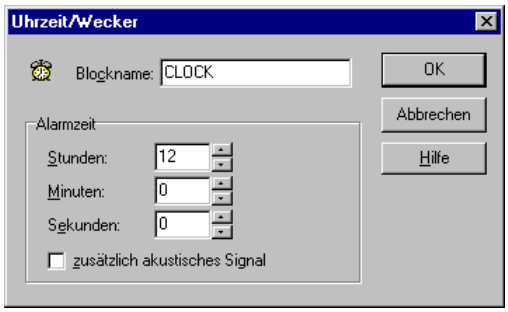
Parameter-dialog:

Uhrzeit/Wecker

Typname: CLOCK

Funktion: Echtzeituhr mit Alarmeinrichtung. An den Ausgängen H, M und S wird die aktuelle Zeit in Stunden, Minuten und Sekunden ausgegeben. Bei Erreichen der voreingestellten Alarmzeit erhält der Alarmausgang A für einen Simulationsschritt High-Pegel. Auf Wunsch kann zusätzlich ein Signalton ausgegeben werden.

Parameterdialog:



Datei (File-Input)

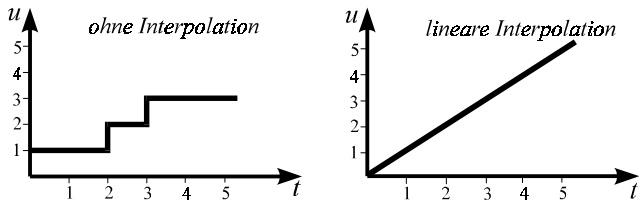
Typname: FILEINPUT

Funktion: Ermöglicht das Einlesen eines Signals $u(t)$ in Form von Wertepaaren (t_i, u_i) aus einer Datei vom Typ SIM. Die Zeitpunkte t_i können beliebig (d. h. nicht zwangsläufig äquidistant), müssen aber in aufsteigender Reihenfolge sortiert sein. Zwischen den eingelesenen Werten kann linear interpoliert werden. Andernfalls wird der letzte gültige Wert jeweils beibehalten, bis ein neuer Wert in der Datei auftritt. In diesem Fall ergibt sich ein stufenförmiger Verlauf der Eingangsgröße.

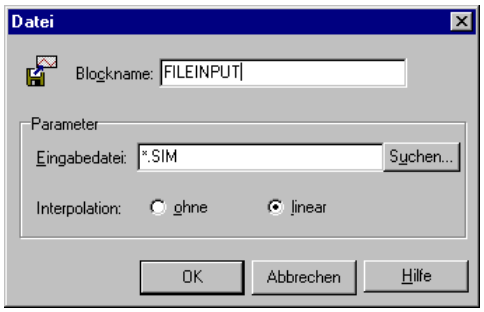
Beispiel: Die eingelesene Datei enthalte die drei Wertepaare

1	1
2	2
3	3.

Für eine Simulation bis zum Zeitpunkt T_{Simu} = 5 ergeben sich dann folgende Signalverläufe:



Parameter-dialog:



Wird für *Eingabedatei* keine Extension angegeben, wird die Extension SIM benutzt. Über *Suchen* kann ein Dateieingabedialog angefordert werden.

ISM-Modul (Eingang) *

Typname: ISM

Funktion: Ermöglicht das Einlesen von Signalen über das ISM-110-Meßmodul der Fa. Gantner. Hinweise dazu entnehmen Sie bitte der zugehörigen Dokumentation.

C-Code (Eingang) *

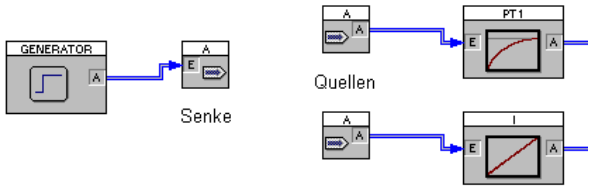
Typname: CCODEINPUT

Funktion: Liefert die Schnittstelle für einen Hardware-Eingang in Verbindung mit dem BORIS-C-Code-Generator. Einzelheiten dazu entnehmen Sie bitte der zugehörigen Dokumentation.

Signalquelle

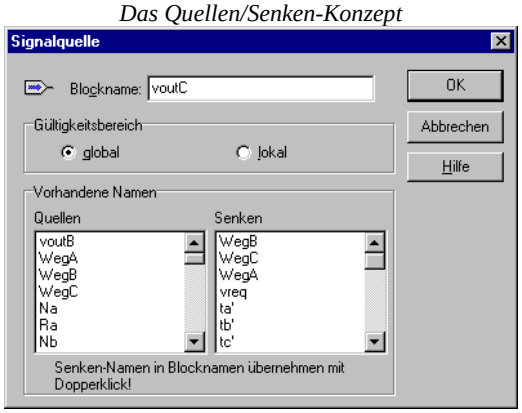
Typname: QUELLE

Funktion: Dieser Block dient zusammen mit dem Ausgangsblocktyp *Signalsenke* zur Realisierung "drahtloser" Verbindungen zwischen Blöcken. Dabei "versendet" eine Signalsenke ihr Eingangssignal unter einem bestimmten Namen (dem Namen des Blocks). Dieses Signal kann dann an beliebigen - auch mehreren - Stellen in der Systemstruktur von einer Signalquelle mit gleichem Namen wieder empfangen werden. Groß- und Kleinschreibung der Blocknamen wird dabei nicht unterschieden. Signalquellen können *lokale* oder *globale* Gültigkeit haben. Während sie im ersten Fall nur innerhalb der zugehörigen System- bzw. Superblockdatei bekannt sind, gelten sie im globalen Fall auch über die Dateigrenzen hinaus. Der Einsatz beider Blocktypen empfiehlt sich insbesondere bei komplexen, stark vermaschten Systemstrukturen, um die Anzahl der sichtbaren Verbindungen gering zu halten.



Das Quellen/Senken-Konzept

Parameterdialog:



In der linken Listbox des Dialogs sind alle bereits vorhandenen Quellen in alphabetischer Reihenfolge aufgelistet, in der rechten Listbox alle Senken. Durch Doppelklick auf einen Senken-Namen in der rechten Listbox kann dieser direkt als Quellen-Name übernommen werden.

Dynamische Blöcke

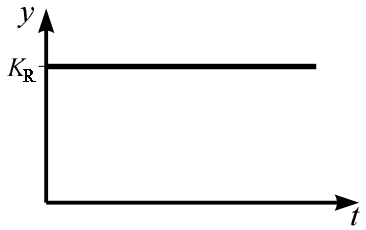
P-Glied

Typname: P

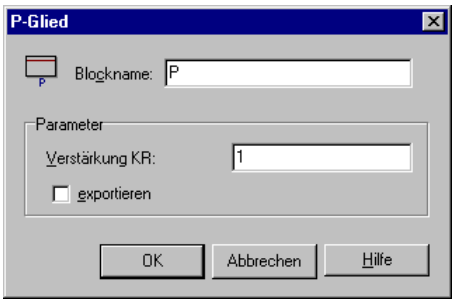
Funktion: Das Proportionalglied (P-Glied) erzeugt aus der Eingangsgröße $x(t)$ eine Ausgangsgröße $y(t)$ gemäß der Beziehung $y(t) = K_R x(t)$. Es besitzt somit die Übertragungsfunktion

$$G(s) = K_R$$

und die nachfolgende Sprungantwort:



Parameter-dialog:



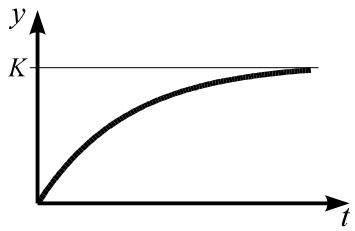
PT_1-Glied

Typname: PT1

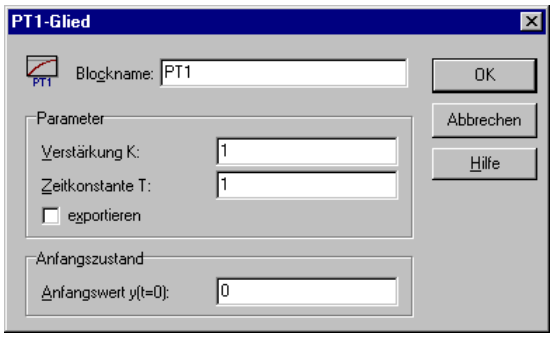
Funktion: Verzögerungsglied 1. Ordnung mit der Verstärkung K und der Zeitkonstanten T. Das PT_1-Glied besitzt demnach die Übertragungsfunktion

$$G(s) = \frac{K}{1+Ts}$$

und die nachfolgende Sprungantwort:



Parameter-dialog:



Für die Simulation muß der Anfangswert $y(t=0)$ vorgegeben werden.

Parameter-grenzen: $T > 0$

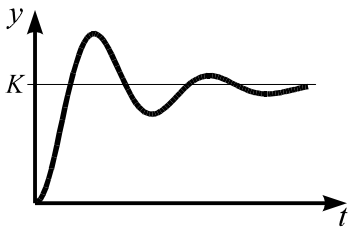
PT_2-Glied (schwingfähig)

Typname: PT2

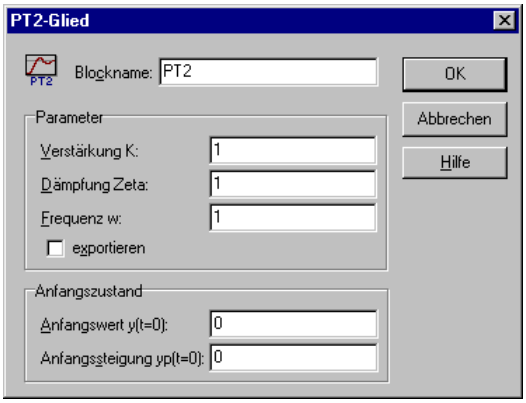
Funktion: Schwingfähiges Verzögerungsglied 2. Ordnung mit der Verstärkung *K*, der Eigenfrequenz ω und der Dämpfung ζ. Das System besitzt die Übertragungsfunktion

$$G(s) = \frac{K}{\left(\frac{s}{\omega}\right)^2 + 2\frac{\zeta}{\omega}s + 1}$$

und nachfolgende Sprungantwort:



Parameter-dialog:



Für die Simulation müssen der Anfangswert $y(t=0)$ und die Anfangssteigung $\dot{y}(t=0)$ vorgegeben werden.

Parameter-grenzen: $\zeta \geq 0,\ \omega > 0$

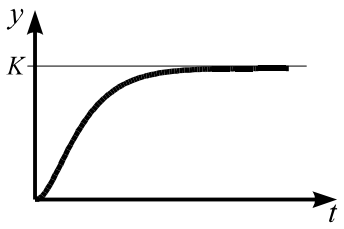
PT_1T_2-Glied

Typname: PT1T2

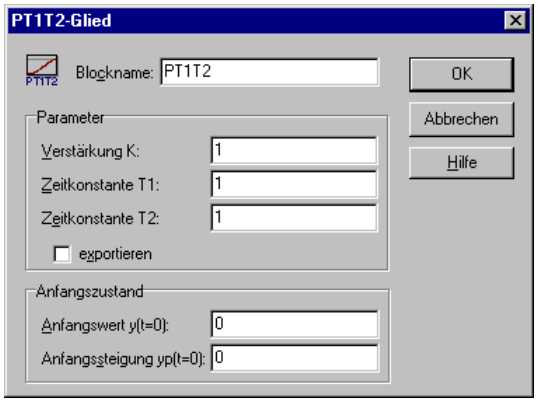
Funktion: Nicht schwingfähiges Verzögerungsglied 2. Ordnung mit der Verstärkung K und den beiden Zeitkonstanten T_1 und T_2. Die Übertragungsfunktion lautet

$$G(s) = \frac{K}{(1+T_1 s)(1+T_2 s)}.$$

Das System besitzt nachfolgende Sprungantwort:



Parameter-dialog:



Für die Simulation müssen der Anfangswert $y(t=0)$ und die Anfangssteigung $\dot{y}(t=0)$ vorgegeben werden.

Parameter-grenzen: $T_1, T_2 > 0$

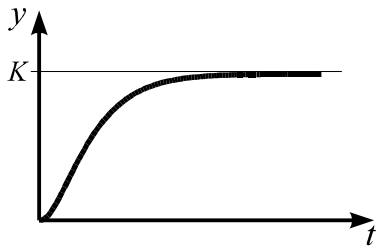
PT_n-Glied

Typname: PTN

Funktion: Verzögerungsglied *n*-ter Ordnung mit der Verstärkung *K* und *n* gleichen Zeitkonstanten *T*. Die zugehörige Übertragungsfunktion lautet

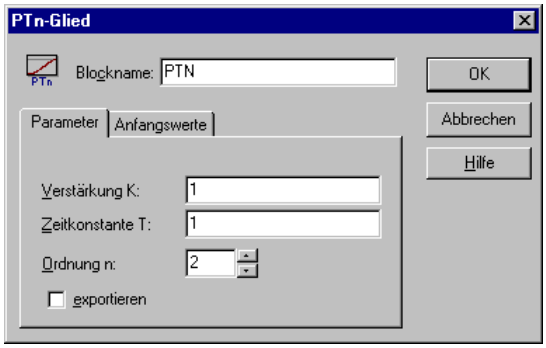
$$G(s) = \frac{K}{(1+Ts)^n}.$$

Das System besitzt die folgende Sprungantwort:



Bei konstantem *T* verläuft die Sprungantwort mit steigendem Wert für *n* zunehmend flacher.

Parameter-dialog:



Parameter-grenzen: $T > 0, \quad 1 \le n \le 8$

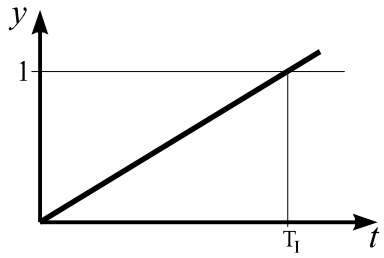
Begrenzter Integrierer

Typname: I

Funktion: Begrenzter Integrierer mit der Integrations-Zeitkonstanten T_I. Innerhalb des linearen Arbeitsbereiches besitzt das System die Übertragungsfunktion

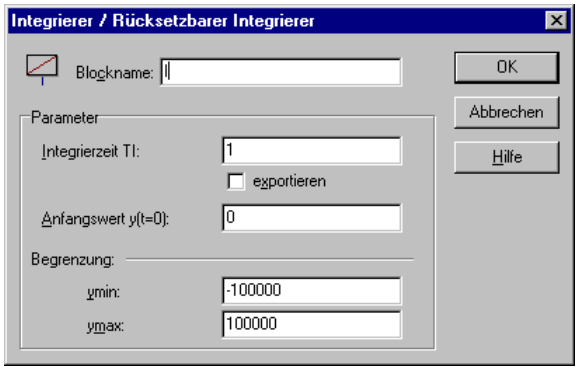
$$G(s) = \frac{1}{T_I s}$$

und nachfolgende Sprungantwort:



Die Ausgangsgröße $y(t)$ des Integrierers wird durch eine Anti-Windup-Halt-Maßnahme auf den Wertebereich $y_{min} \le y(t) \le y_{max}$ begrenzt. Diese hält die Integration an, solange die Ausgangsgröße an der oberen Begrenzung y_{max} ist und die Eingangsgröße des Integrierers positiv bzw. die Ausgangsgröße an der unteren Begrenzung y_{min} und die Eingangsgröße negativ ist. Bei Vorzeichenwechsel der Eingangsgröße löst sich die Ausgangsgröße dann jeweils unmittelbar von der Begrenzung.

Parameter-dialog:



Zur Simulation muß der Anfangswert $y(t=0)$ vorgegeben werden.

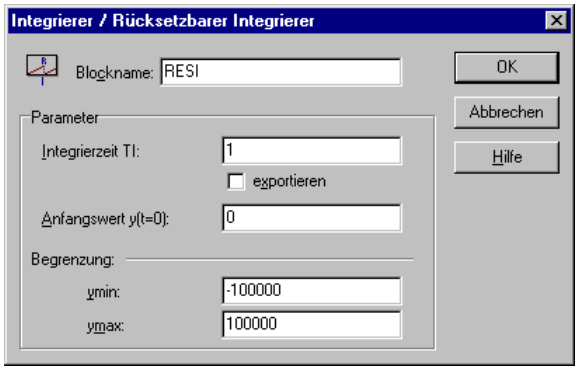
Parameter-grenzen: $T_I > 0$

Rücksetzbarer Integrierer

Typname: RESI

Funktion: Der Block entspricht dem begrenzten Integrierer, die Ausgangsgröße kann während der Simulation jedoch durch eine positive Flanke am Steuereingang R jederzeit auf den Anfangswert zurückgesetzt werden.

Parameter-dialog:



Parameter-grenzen: $T_I > 0$

Differenzierer

Typname: D

Funktion: Idealer Differenzierer mit der Differentiations-Zeitkonstanten T_D , der Übertragungsfunktion

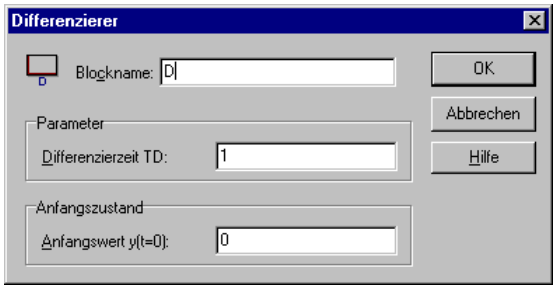
$$G(s) = T_D s$$

und nachfolgender Sprungantwort:

Die Ausgangsgröße nimmt zum Zeitpunkt $t = 0$ theoretisch den Wert ∞ an. In der Simulation wird der Ausgangswert bei einem Eingangssprung der Höhe Δx und der Simulationsschrittweite ΔT begrenzt auf

$$y_{max} = \frac{\Delta x}{\Delta T}.$$

Parameter-dialog:



Zur Simulation muß der Anfangswert $y(t=0)$ des Differenzierers vorgegeben werden.

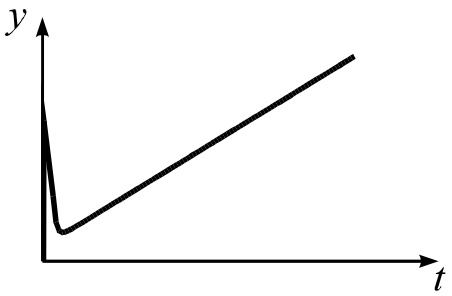
PID-Regler

Typname: PID

Funktion: PID-Regler mit der Verstärkung K_R, der Nachstellzeit T_N und der Vorhaltezeit T_V. Der D-Anteil des Reglers ist mit einem zusätzlichen PT_1-Glied versehen (verzögerte Differentiation). Die Ausgangsgröße $y(t)$ wird durch eine Anti-Windup-Halt-Maßnahme auf den Wertebereich $y_{min} \le y(t) \le y_{max}$ beschränkt. Im linearen Arbeitsbereich weist der PID-Regler die Übertragungsfunktion

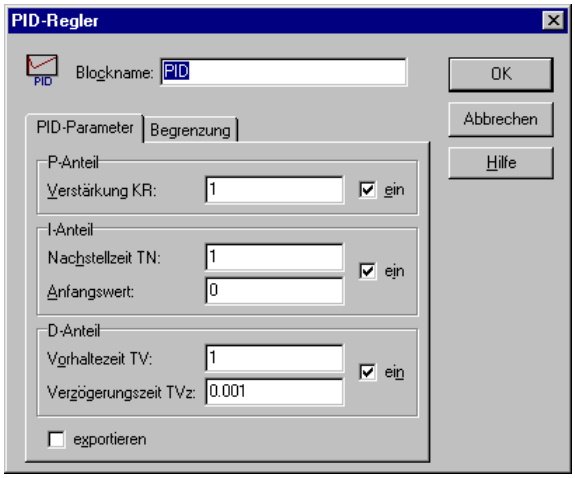
$$G(s) = K_R\left(1 + \frac{1}{T_N s} + \frac{T_V s}{1 + T_{Vz} s}\right)$$

und folgende Sprungantwort auf:



P-, I- und D-Anteil sind getrennt voneinander zu- und abschaltbar. Ebenso kann die Begrenzung zu- oder abgeschaltet werden. Für die Simulation muß der Anfangswert des Integrierers vorgegeben werden.

Parameter-dialog:



Parameter-grenzen: $T_N, T_{Vz} > 0, \quad T_V \geq 0$

Adaptiver PID-Regler mit steuerbarer Begrenzung

Typname: ADAPID

Funktion: Der Block entspricht in seiner Funktionsweise dem zuvor beschriebenen PID-Regler, die Parameter K_R, T_N und T_V können jedoch während der Simulation über die Steuereingänge P, I und D modifiziert werden. Die Modifikation kann multiplikativ oder additiv erfolgen. Die aktuellen Werte für die Parameter ergeben sich aus den über den Parameterdialog eingestellten Grundwerten K_{R0}, T_{N0} und T_{V0} wie folgt:

Betriebsart *multiplikativ*:

$$K_R = K_{R0} x_P(t)$$
$$T_N = T_{N0} x_I(t)$$
$$T_V = T_{V0} x_D(t)$$

Betriebsart *additiv*:

$$K_R = K_{R0} + x_P(t)$$
$$T_N = T_{N0} + x_I(t)$$
$$T_V = T_{V0} + x_D(t)$$

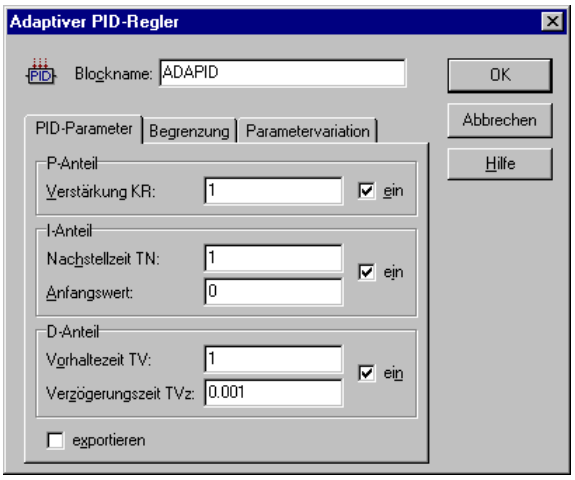
$x_P(t), x_I(t)$ und $x_D(t)$ sind die an den Steuereingängen P, I und D anliegenden Signale. In der Betriebsart multiplikativ werden offene Steuereingänge zu eins, in der Betriebsart additiv zu null gesetzt.

Über den Steuereingang S kann darüber hinaus die Begrenzung des Reglers von außen beeinflußt werden. Ist der Steuereingang offen, so arbeitet der Regler mit der festen Begrenzung der Ausgangsgröße auf den Bereich $y_{min} \le y(t) \le y_{max}$. Ist der Steuereingang angeschlossen, so ergibt sich die tatsächliche Begrenzung durch Multiplikation der über den Dialog eingestellten Werte mit dem am Steuereingang anliegenden Signal $x_S(t)$:

$$y_{min,\,akt}(t) = y_{min} \cdot x_S(t)$$
$$y_{max,\,akt}(t) = y_{max} \cdot x_S(t)$$

Voraussetzung ist natürlich jeweils, daß die Begrenzung auch aktiviert wurde!

Parameterdialog:



Parametergrenzen: $T_N, T_{Vz} > 0,\ T_V \ge 0$

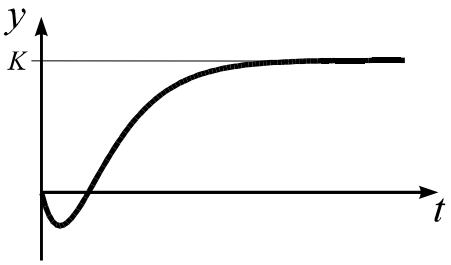
Allpaß Typ I

Typname: ALLPASS_1

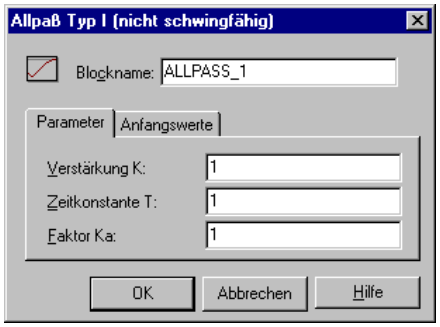
Funktion: Allpaßsystem 2. Ordnung (nicht schwingfähig) mit der Übertragungsfunktion

$$G(s) = K\frac{1 - K_a Ts}{(1 + K_a Ts)(1 + Ts)}$$

und nachfolgender Sprungantwort:



Parameter-dialog:



Parameter-grenzen: $T,\ K_a > 0$

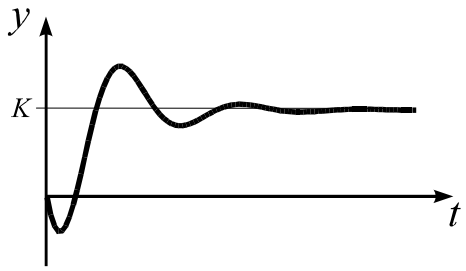
Allpaß Typ II

Typname: ALLPASS_2

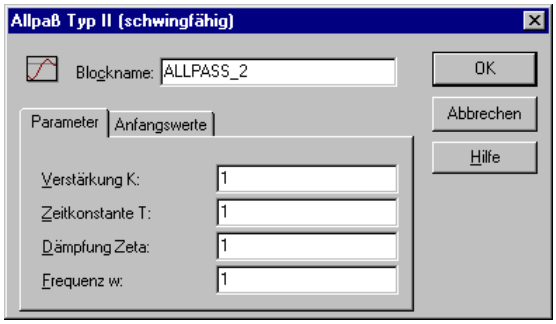
Funktion: Allpaßsystem 2. Ordnung (schwingfähig) mit der Übertragungsfunktion

$$G(s) = K\frac{1-Ts}{\left(\frac{s}{\omega}\right)^2 + 2\frac{\zeta}{\omega}s + 1}$$

und der nachfolgenden Sprungantwort:



Parameter-dialog:



Parameter-grenzen: $T,\ \omega > 0,\ \zeta \geq 0$

Totzeitglied

Typname: TOTZEIT

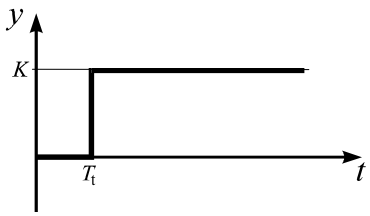
Funktion: Das Totzeitglied verzögert die Eingangsgröße $x(t)$ um die Totzeit T_t gemäß

$$y(t) = \begin{cases} y_0 & \text{für } t < T_t \\ K\,x(t - T_t) & \text{für } t \geq T_t \end{cases}$$

und besitzt die Übertragungsfunktion

$$G(s) = Ke^{-T_t s}$$

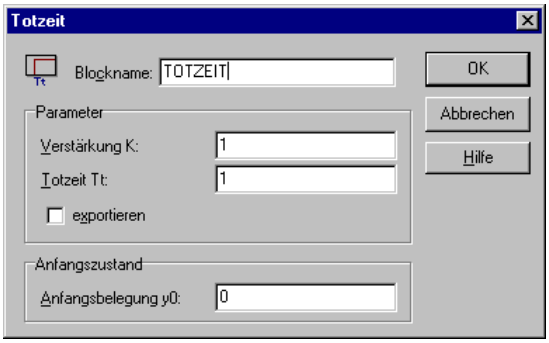
sowie folgende Sprungantwort:



Der interne Speicher des Totzeitgliedes kann beliebig viele Elemente aufnehmen. Die Totzeit sollte möglichst ein ganzzahliges Vielfaches der Simulationsschrittweite ΔT betragen.

Hinweis: Ein Systemblocktyp mit *variabler* Totzeit befindet sich unter dem Namen VDELAY32.DLL im Verzeichnis *UserDLLs*.

Parameterdialog:



Parametergrenzen: $T_t \geq 0$

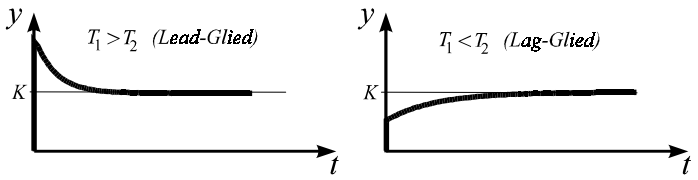
Lead/Lag-Glied

Typname: LEADLAG

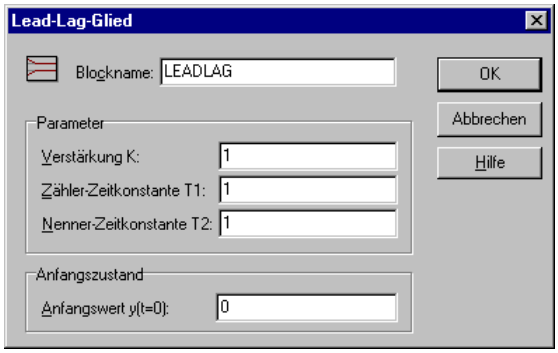
Funktion: Rationales Glied 1. Ordnung mit der Übertragungsfunktion

$$G(s) = K\frac{1+T_1 s}{1+T_2 s}$$

und den folgenden Sprungantworten:



Parameter-dialog:



Für die Simulation muß der Anfangswert $y(t=0)$ vorgegeben werden.

Parameter-grenzen: $T_1 \geq 0, \quad T_2 > 0$

Vorhalteglied (DT_1-Glied)

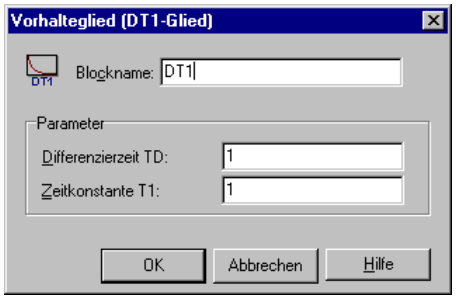
Typname: DT1

Funktion: Rationales Glied 1. Ordnung (verzögerter Differenzierer) mit der Übertragungsfunktion

$$G(s) = \frac{T_D s}{1 + T_1 s}$$

und der folgenden Sprungantwort:

Parameterdialog:



Parametergrenzen: $T_D,\ T_1 > 0$

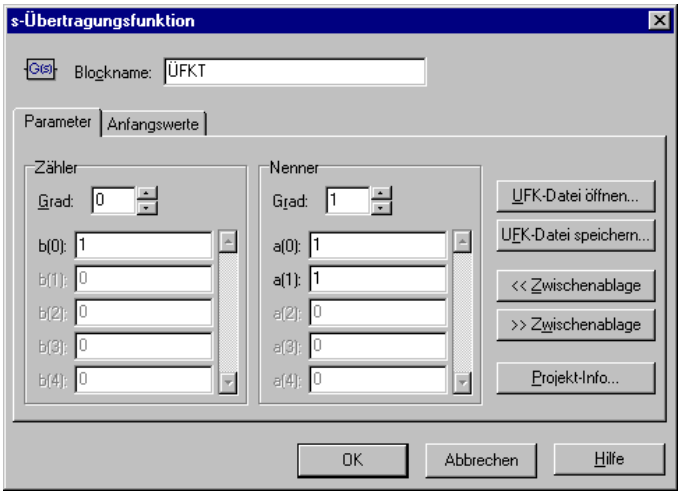
Übertragungsfunktion

Typname: ÜFKT

Funktion: Frei parametrierbare *s*-Übertragungsfunktion der Form

$$G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0}, \quad m \le n\ .$$

Parameter-dialog:



Die Übertragungsfunktion kann wahlweise über Tastatur eingegeben oder aus der Zwischenablage bzw. einer UFK-Datei geholt werden.

Parameter-grenzen: $m \le n,\ n \le 8$

$a_n \neq 0$

Dgl.-System

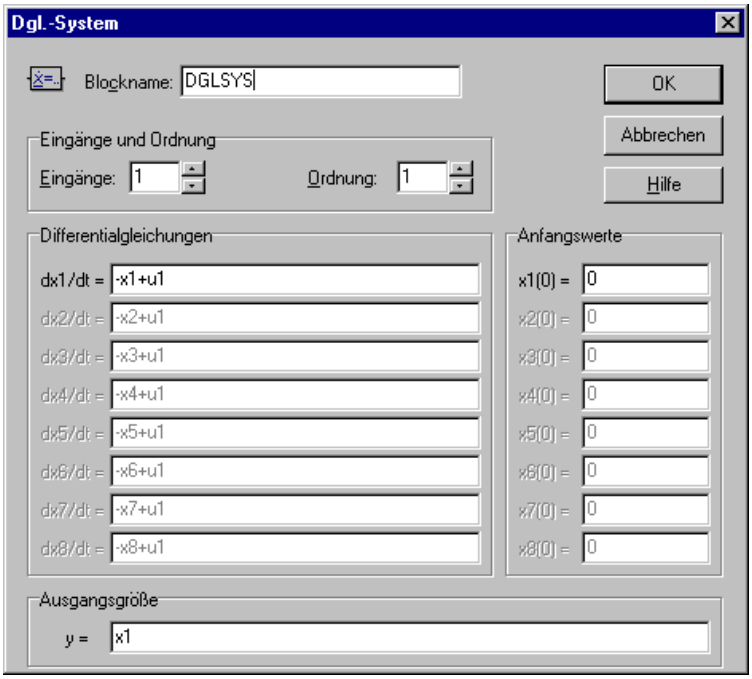
Typname: DGLSYS

Funktion: Frei parametrierbares Dgl.-System der Form

$$\begin{aligned} \dot{\underline{x}} &= F(\underline{x}, \underline{u}) \\ y &= g(\underline{x}, \underline{u}) \end{aligned} .$$

Das Dgl.-System kann sowohl linear als auch nichtlinear sein und wird über einen Funktionsparser interpretiert. $\underline{x}(t)$ ist der Zustandsvektor des Systems (maximale Ordnung: 8), $\underline{u}(t)$ der Eingangsvektor und $y(t)$ die Ausgangsgröße. Der Anfangswertvektor $\underline{x}(t = 0)$ kann ebenfalls frei vorgegeben werden.

Parameter-dialog:



Die Zustandsgrößen werden mit *x1*, *x2*, ... bezeichnet, die Eingangsgrößen mit *u1*, *u2*, ... Obiger Dialog enthält somit das - in diesem Fall lineare - Dgl.-System

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = x_3$$
$$\dot{x}_3 = -x_1 - 2x_2 - 3x_3 + u_1$$
$$y = x_1$$

mit den Anfangswerten $\underline{x}(0) = \underline{0}$.

Einheitsverzögerung

Typname: UNITDELAY

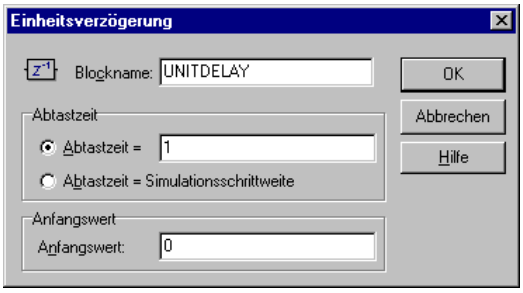
Funktion: Dieser Block realisiert ein Abtast-/Halteglied mit einer Verzögerung um eine Abtastperiode *T*. Er besitzt somit die *z*-Übertragungsfunktion

$$G(z) = \frac{1}{z}.$$

Die Abtastzeit *T* und der Anfangswert zu Beginn der Simulation sind frei wählbar.

Dieser Blocktyp kann sinnvollerweise auch zur Verhinderung *algebraischer Schleifen* eingesetzt werden, um innerhalb einer Rückführung eine Verzögerung von genau einem Takt zu erreichen. Damit in solchen Fällen bei einer Änderung der Simulationsschrittweite nicht jedesmal die Abtastzeit angepaßt werden muß, kann diese über den Parameterdialog direkt an die Abtastzeit angekoppelt werden.

Parameterdialog:



Die Abtastzeit sollte ein ganzzahliges Vielfaches der Simulationsschrittweite betragen, andernfalls erfolgt zu Beginn der Simulation eine Warnmeldung.

Parametergrenzen: $T > 0$

z-Übertragungsfunktion

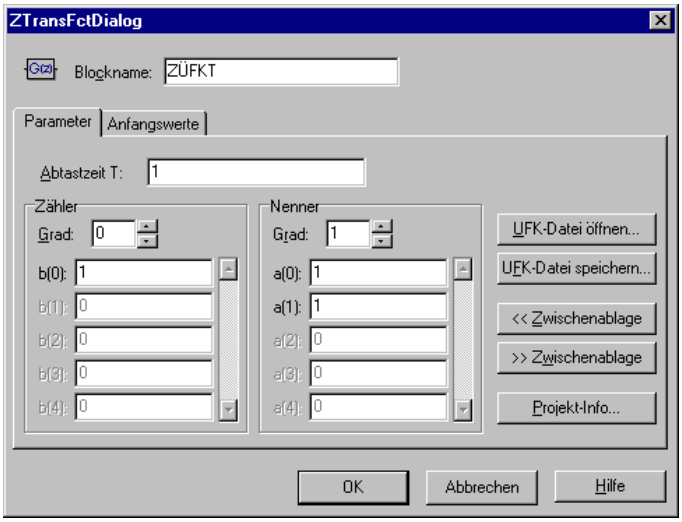
Typname: ZÜFKT

Funktion: Frei parametrierbare *z*-Übertragungsfunktion der Form

$$H(z) = \frac{b_m z^m + b_{m-1} z^{m-1} + \ldots + b_1 z + b_0}{a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0}, \quad m \le n \; .$$

Die Abtastzeit *T* ist frei wählbar.

Parameter-dialog:



Die Übertragungsfunktion kann wahlweise über Tastatur eingegeben oder aus der Zwischenablage bzw. einer UFK-Datei geholt werden. Das Dateiformat entspricht dem der *s*-Übertragungsfunktion, an die Stelle der Totzeit tritt hier jedoch die Abtastzeit. Die Abtastzeit sollte ein ganzzahliges Vielfaches der Simulationsschrittweite betragen, andernfalls erfolgt zu Beginn der Simulation eine Warnmeldung.

Parameter-grenzen: $m \le n,\ \ n \le 8,\ \ a_n \neq 0$

Statische Blöcke

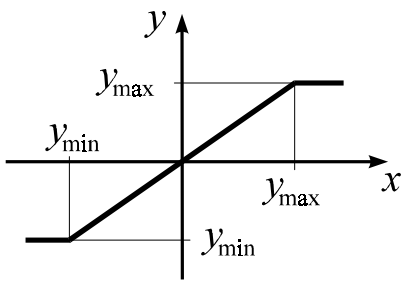
Begrenzer

Typname: BEGRENZER

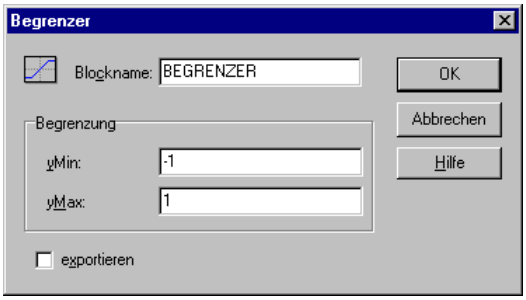
Funktion: Die Begrenzer- oder Sättigungskennlinie begrenzt die Ausgangsgröße auf den Wertebereich $y_{\min} \le y(t) \le y_{\max}$ gemäß der Vorschrift

$$y(t) = \begin{cases} y_{\min} & \text{für } x(t) < y_{\min} \\ x(t) & \text{für } y_{\min} \le x(t) \le y_{\max} \\ y_{\max} & \text{für } x(t) > y_{\max} \end{cases} .$$

Im linearen Arbeitsbereich besitzt das Kennlinienglied die Steigung 1. Die Kennlinie hat demnach die folgende Gestalt:



Parameter-dialog:



Parameter-grenzen: $y_{\min} \le y_{\max}$

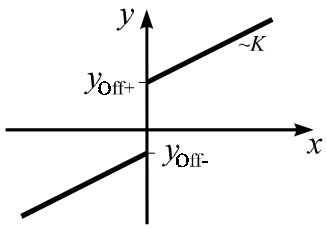
Vorlastkennlinie

Typname: VORLAST

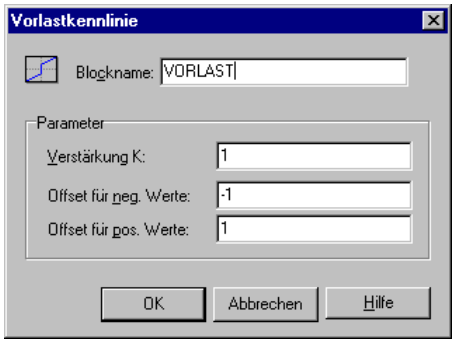
Funktion: Die Vorlastkennlinie ist stückweise linear mit einer Unstetigkeit an der Stelle $x = 0$. Sie wird charakterisiert durch die Beziehung

$$y(t) = \begin{cases} y_{Off-} + K\,x(t) & \text{für} \quad x(t) \le 0 \\ y_{Off+} + K\,x(t) & \text{für} \quad x(t) > 0 \end{cases}$$

und hat die nachfolgende Gestalt:



Parameter-dialog:



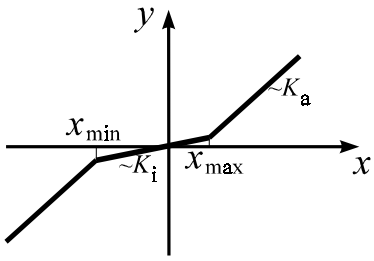
Unempfindlichkeitszone

Typname: UNEMPFINDLICH

Funktion: Dieser Systemtyp realisiert eine Verringerung der Verstärkung von K_a auf K_i innerhalb einer Unempfindlichkeitszone $[\,x_{min}, x_{max}\,]$ der Eingangsgröße $x(t)$ gemäß der Beziehung

$$y(t) = \begin{cases} K_a\,x(t) + x_{min}\,(K_i - K_a) & \text{für } x(t) < x_{min} \\ K_i\,x(t) & \text{für } x_{min} \le x(t) \le x_{max} \\ K_a\,x(t) + x_{max}\,(K_i - K_a) & \text{für } x(t) > x_{max} \end{cases} .$$

Die Kennlinie hat die folgende Gestalt:



Für $K_i = 0$ erhält man eine Kennlinie mit toter Zone.

Parameter-dialog:

Parameter-grenzen: $x_{min} \le x_{max}$

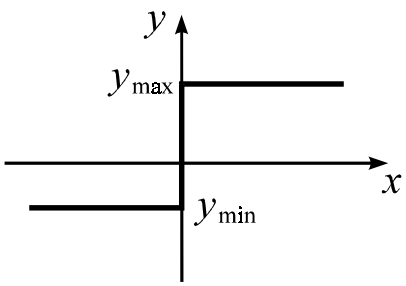
Zweipunktkennlinie

Typname: ZWEIPUNKT

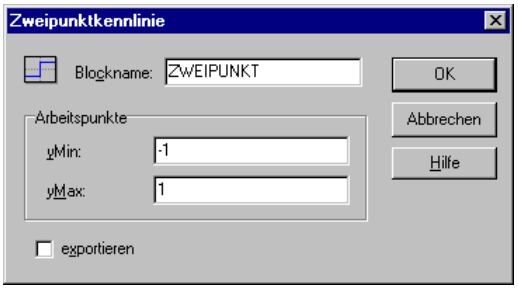
Funktion: Die Kennlinie gehorcht der Beziehung

$$y(t) = \begin{cases} y_{min} & \text{für } x(t) < 0 \\ y_{max} & \text{für } x(t) \ge 0 \end{cases}$$

und hat die folgende Gestalt:



Parameter-dialog:



Parameter-grenzen: $y_{\min} \le y_{\max}$

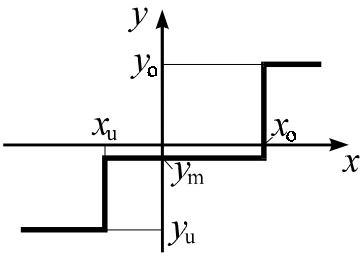
Dreipunktkennlinie

Typname: DREIPUNKT

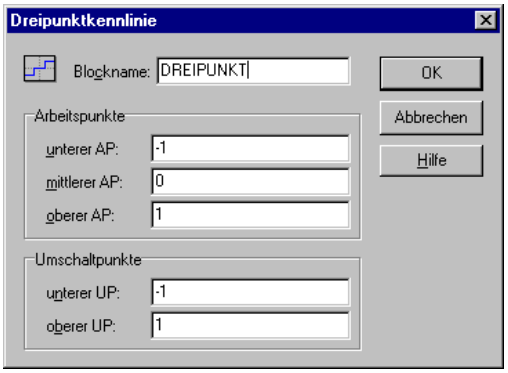
Funktion: Die Kennlinie gehorcht der Beziehung

$$y(t) = \begin{cases} y_u & \textit{für} \quad x(t) < x_u \\ y_m & \textit{für} \quad x_u \le x(t) \le x_o \\ y_o & \textit{für} \quad x(t) > x_o \end{cases}$$

und hat die folgende Gestalt:



Parameter-dialog:

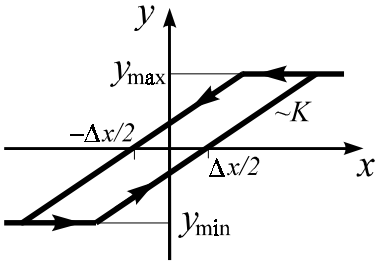


Parameter-grenzen: $x_u \le x_o, y_u \le y_m \le y_o$

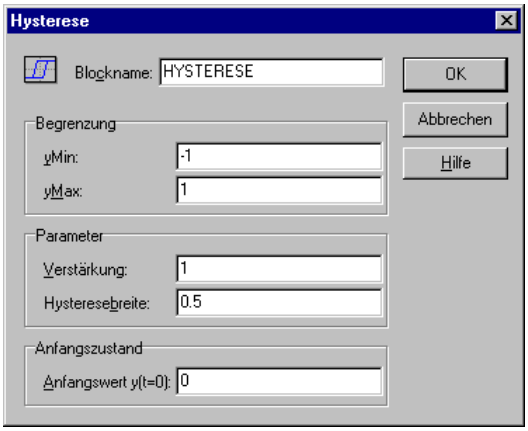
Hysteresekennlinie

Typname: HYSTERESE

Funktion: Die Hysteresekennlinie wird durch Verstärkung K, Hysteresebreite Δx und Begrenzung y_{min} bzw. y_{max} charakterisiert und besitzt folgende Gestalt:



Parameter-dialog:



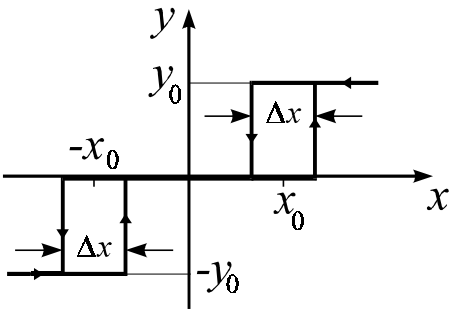
Zur Simulation muß der Anfangswert $y(t=0)$ vorgegeben werden.

Parameter-grenzen: $K, \Delta x > 0,\ y_{\min} \le y_{\max}$

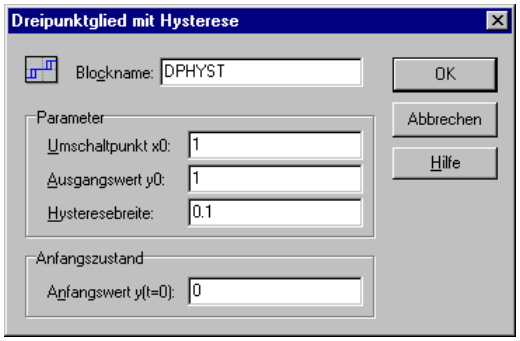
Dreipunktglied mit Hysterese

Typname: DPHYST

Funktion: Stellt eine Kombination aus einem symmetrischen Dreipunktglied mit den Umschaltpunkten x_0 und $-x_0$ und den Arbeitspunkten y_0 und $-y_0$ und einem Hystereseglied mit der Hysteresebreite Δx dar.



Parameter-dialog:



Parameter-grenzen: $x_0, y_0, \Delta x > 0$

Benutzerdefinierte Kennlinie

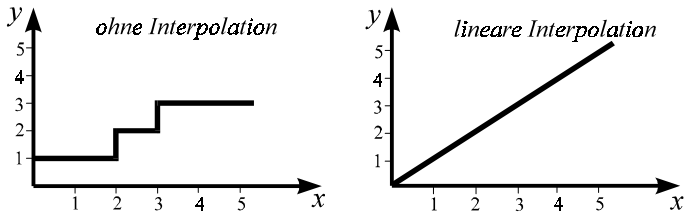
Typname: KENNLINIE

Funktion: Ermöglicht die Definition einer Kennlinie $y(x)$ über n Wertepaare (x_i, y_i). Die Stützstellen x_i können beliebig (d. h. nicht zwangsläufig äquidistant), müssen aber in aufsteigender Reihenfolge sortiert sein. Zwischen den eingelesenen Werten kann linear interpoliert werden. Andernfalls wird der letzte gültige Wert jeweils beibehalten, bis ein neuer Wert auftritt. In diesem Fall ergibt sich eine stufenförmige Kennlinie.

Beispiel: Die Kennlinie sei definiert über die drei Wertepaare

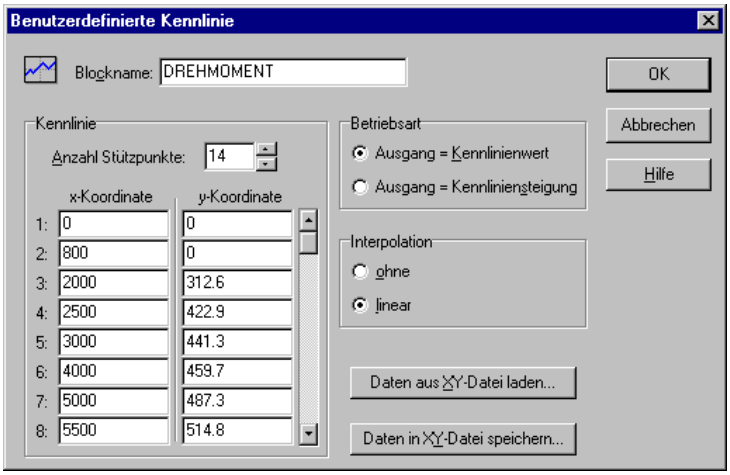
(1, 1), (2, 2), (3, 3).

Es ergeben sich dann folgende Kennlinienverläufe:



Statt des Ordinatenwertes selbst kann auch die *Steigung* $\mathrm{d}y/\mathrm{d}x$ der Kennlinie für den am Eingang anliegenden Wert ermittelt und ausgegeben werden. In diesem Fall ist eine ggf. zusätzlich gewählte Interpolation ohne Wirkung. Für obige Beispielkennlinie ergäbe sich z. B. für alle Eingangswerte eine konstante Steigung von eins.

Parameter-dialog:



Parameter-grenzen: $2 \le n \le 1000$

Stellglieder

Stellglied Typ I

Typname: STELLGLIED_1

Funktion: Dieser Block stellt ein Stellglied dar, das neben der typischen Ausgangsgrößen-begrenzung

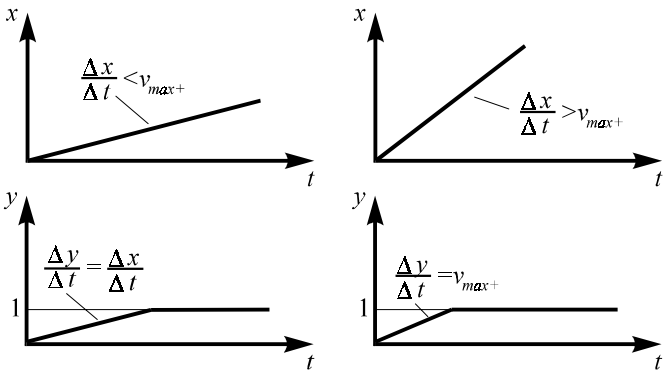
$$y(t) = \begin{cases} y_{\min} & \text{für } x(t) < y_{\min} \\ x(t) & \text{für } y_{\min} \le x(t) \le y_{\max} \\ y_{\max} & \text{für } x(t) > y_{\max} \end{cases}$$

(vgl. Begrenzer-Block) eine Anstiegsgeschwindigkeitsbegrenzung besitzt. Die Änderungsgeschwindigkeit der Ausgangsgröße des Stellglieds wird dabei für negative Änderungen auf $v_{\text{max-}}$ und für positive Änderungen auf $v_{\text{max+}}$ begrenzt:

$$v_{\text{max}-} \le \frac{\mathrm{d}y}{\mathrm{d}t} \le v_{\text{max}+}$$

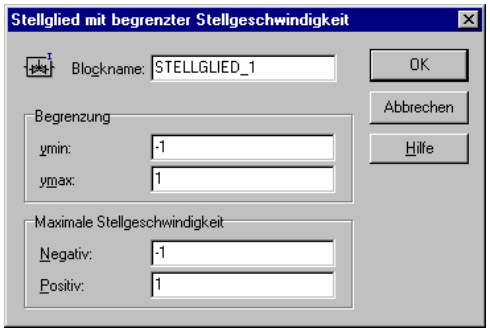
Nachfolgendes Bild zeigt den Verlauf der Ausgangsgröße bei verschiedenen Eingangsrampen für den Fall

$$y_{\min} = -1, y_{\max} = 1, \quad v_{\text{max}-} = -1, v_{\text{max}+} = 1 .$$



Verhalten des Stellglieds für eine Eingangsrampe mit kleiner (links) und großer Steigung, d. h. Änderungsgeschwindigkeit (rechts)

Parameterdialog:



Parameter-grenzen: $y_{\min} < y_{\max}$, $v_{\max-} < 0, v_{\max+} > 0$

Stellglied Typ II

Typname: STELLGLIED_2

Funktion: Dieser Stellglied-Typ weist wie Typ I eine Ausgangsgrößenbegrenzung auf, besitzt jedoch eine *konstante* Anstiegsgeschwindigkeit: Für negative Eingangsgrößenänderungen ändert sich die Ausgangsgröße unabhängig vom Betrag der Eingangsgrößenänderung mit v_-, für positive Eingangsgrößenänderungen mit v_+. Außerdem besitzt das Stellglied eine tote Zone der Breite Δx. Eingangsgrößen, die betragsmäßig kleiner sind als $\Delta x / 2$, werden ignoriert, d. h. der aktuelle Ausgangswert wird beibehalten.

Parameter-dialog:

Parameter-grenzen: $y_{\min} < y_{\max}$
$v_- < 0, \quad v_+ > 0$
$\Delta x / 2 \geq 0$

Funktionsblöcke

Verknüpfer

Typname: VERKNUEPFER

Funktion: Dieser Block erlaubt die Verknüpfung von zwei bis 50 Eingangsgrößen über die Operationen Summation, Multiplikation oder Division. Für jede Eingangsgröße x_i kann getrennt eine Vorzeichenumkehr erfolgen. Es gilt:

Betriebsart *Summation*: $y = \pm x_1 \pm x_2 \pm x_3 \pm \ldots$

Betriebsart *Multiplikation*: $y = (\pm x_1) \cdot (\pm x_2) \cdot (\pm x_3) \cdot \ldots$

Betriebsart *Division*: $y = (\pm x_1) / (\pm x_2) / (\pm x_3) / \ldots$

In den Betriebsarten *Multiplikation* und *Division* werden etwaige offene Eingänge zu eins gesetzt. In der Betriebsart *Summation* werden die Block-Eingangsfelder mit dem jeweiligen Vorzeichen der Eingangsgröße gekennzeichnet.

Parameterdialog:

Funktion einer Veränderlichen

Typname: FKT1

Funktion: Dieser Block erlaubt die Definition einer Funktion $y = f(x)$. Zur Auswahl stehen folgende Funktionen:

$y = \sin(x)$ $y = \cos(x)$ $y = \tan(x)$ $y = \exp(x)$ $y = \ln(x)$

$y = |x|$ $y = x^2$ $y = x^3$ $y = \sqrt{x}$

Alternativ dazu kann die Funktion vom Anwender frei definiert und über einen Funktionsparser interpretiert werden (siehe dazu Blockbeschreibung *Generator*). Als unabhängige Variable ist in diesem Fall "x" anzugeben.

Beispiel: *atan(x)+asin(x)+5*

Parameterdialog:

Funktion zweier Veränderlicher

Typname: FKT2

Funktion: Dieser Block erlaubt die Definition einer Funktion $z = f(x, y)$, wobei x und y Eingangsgrößen des Blocks und z seine Ausgangsgröße ist. Zur Auswahl stehen folgende Funktionen:

$z = x + y$ $z = x - y$ $z = xy$ $z = x / y$ $z = x^y$ $z = x^{1/y}$

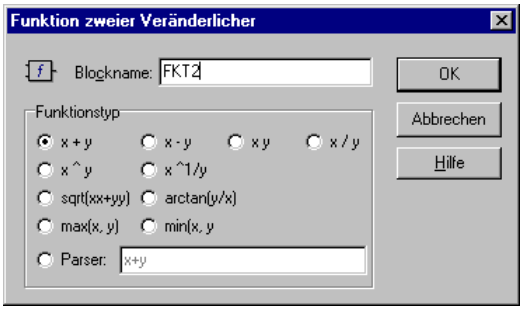
$z = \sqrt{x^2 + y^2}$ $z = \arctan(y / x)$ $z = \max(x, y)$ $z = \min(x, y)$

Alternativ dazu kann die Funktion vom Anwender frei definiert und über einen Funktionsparser interpretiert werden (siehe dazu Blockbeschreibung *Generator*). Als unabhängige Variablen sind in diesem Fall "x" und "y" anzugeben.

Beispiel: *sin(x) + cos(y)*

Die Interpretation der Funktion über den Parser kann je nach Komplexität der Funktion sehr rechenzeitaufwendig sein.

Parameterdialog:



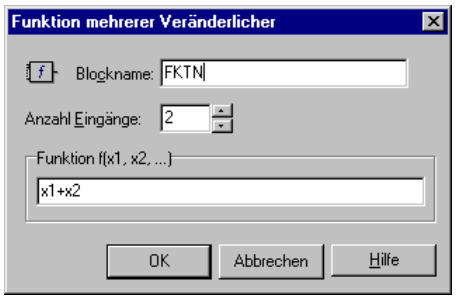
Funktion mehrerer Veränderlicher

Typname: FKTN

Funktion: Dieser Block erlaubt die Definition einer Funktion mit bis zu 50 Veränderlichen über einen Funktionsparser (siehe dazu Blockbeschreibung *Generator*). Als unabhängige Variablen sind in diesem Fall "x1" bis "x10" anzugeben.

Beispiel: $x1 + x2 + x3 - x4 * x5$

Parameterdialog:



Extremwertbestimmung

Typname: EXTREMWERT

Funktion: Dieser Block erlaubt die Bestimmung des Extremwerts der Eingangsgröße $x(t)$. Es sind vier verschiedene Betriebsarten möglich:

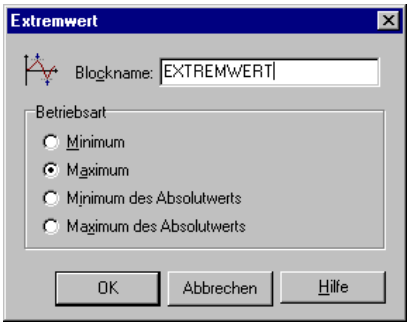
Betriebsart *Minimum*: $y(t) = \min_{0<t<T_{Simu}} x(t)$

Betriebsart *Maximum*: $y(t) = \max_{0<t<T_{Simu}} x(t)$

Betriebsart *Betragsminimum*: $y(t) = \min_{0<t<T_{Simu}} |x(t)|$

Betriebsart *Betragsmaximum*: $y(t) = \max_{0<t<T_{Simu}} |x(t)|$

Parameter-dialog:

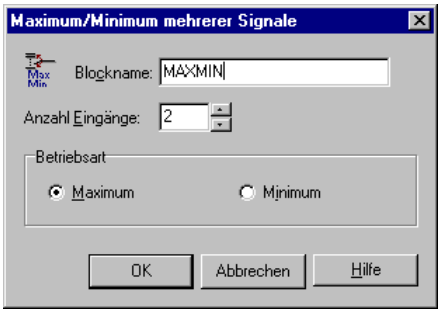


Minimum-/Maximumbestimmung

Typname: MAXMIN

Funktion: Dieser Block schaltet entweder das Minimum oder das Maximum aller Eingangssignale auf den Ausgang durch.

Parameter-dialog:



Statistikfunktionen

Typname: STATISTIK

Funktion: Dieser Block erlaubt die Berechnung statistischer Kennwerte der Eingangsgröße $x(t)$. Als Statistikwerte bezogen auf alle bis zum *k*-ten Simulationsschritt anliegenden Werte sind verfügbar:

Mittelwert: $$y(t_k) = \frac{1}{k}\sum_{i=1}^{k} x(t_i)$$

Mittelwert des Betrags: $$y(t_k) = \frac{1}{k}\sum_{i=1}^{k} \left|x(t_i)\right|$$

Standardabweichung: $$y(t_k) = \sqrt{\frac{1}{k}\left(\sum_{i=1}^{k} x^2(t_i) - \frac{1}{k}\left(\sum_{i=1}^{k} x(t_i)\right)^2\right)}$$

Weiterhin sind verschiedene *gleitende* Funktionen verfügbar, die sich nur auf ein bestimmtes *Zeitfenster* (Zeitspanne) beziehen, das die letzten *n* Werte umfaßt. Wird beispielsweise eine Zeitspanne von 5 bei einer Simulationsschrittweite von 0.1 vorgegeben, so bestimmen sich die gleitenden Kennwerte jeweils aus den letzten 5/0.1 = 50 Eingangswerten. Folgende Kennwerte sind verfügbar:

Gleitender Mittelwert: $$y(t_k) = \frac{1}{n} \sum_{i=k-n+1}^{k} x(t_i)$$

Gleitende Standardabweichung:

$$y(t_k) = \sqrt{\frac{1}{n}\left(\sum_{i=k-n+1}^{k} x^2(t_i) - \frac{1}{n}\left(\sum_{i=k-n+1}^{k} x(t_i) \right)^2 \right)}$$

Gleitende Summe: $$y(t_k) = \sum_{i=k-n+1}^{k} x(t_i)$$

Gleitende Summe der Quadrate: $$y(t_k) = \sum_{i=k-n+1}^{k} x^2(t_i)$$

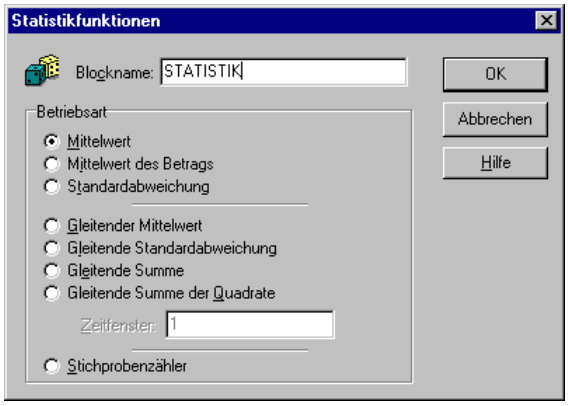
Schließlich kann der Block auch als *Stichprobenzähler* betrieben werden:

Stichprobenzähler: $y(t_k) = k$

Hinweis: Die Gleichungen für die gleitenden Kennwerte gelten erst im "eingeschwungenen" Zustand: Zu Beginn der Simulation - solange noch keine n Werte zur Verfügung stehen, also während der ersten n-1 Schritte - werden alle bis dahin vorliegenden Eingangswerte zur Ermittlung der gleitenden Kenngrößen benutzt! Man beachte weiterhin, daß in allen Betriebsarten der Eingangswert zum Zeitpunkt $t = 0$ mitgerechnet wird! Nach dem ersten "echten" Simulationsschritt ist also $k = 2$!

Der Block kann über eine positive Flanke am Rücksetzeingang R jederzeit zurückgesetzt werden. Die Berechnung der statistischen Kennwerte erfolgt dann ab diesem Zeitpunkt neu.

Parameter-dialog:



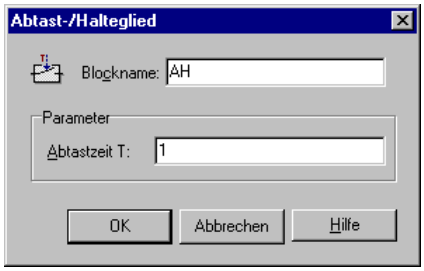
Parameter-grenzen: Das Zeitfenster darf maximal 1000 zurückliegende Werte umfassen.

Abtast-/Halteglied

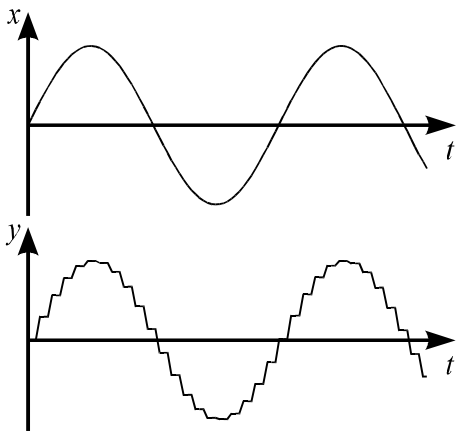
Typname: AH

Funktion: Dieser Funktionsblock realisiert ein Abtast-Halteglied nullter Ordnung: Die Eingangsgröße wird in einem vorgebbaren Zeitabstand T abgetastet, auf den Ausgang geschaltet und dort bis zum nächsten Abtastzeitpunkt konstantgehalten. Die Abtastzeit T sollte ein ganzzahliges Vielfaches der Simulationsschrittweite ΔT betragen.

Parameter-dialog:



Beispiel: Nachfolgende Grafik zeigt den Ausgangsgrößenverlauf eines Abtast-/Haltegliedes bei sinusförmiger Eingangsgröße.



Parameter-grenzen: $T > 0$

Trigerbares Abtast-/Halteglied

Typname: EXTERNAH

Funktion: Dieser Funktionsblock realisiert ein Abtast-Halteglied nullter Ordnung (Analogspeicher), bei dem der Abtastzeitpunkt durch eine positive Flanke am Takteingang C vorgegeben wird.

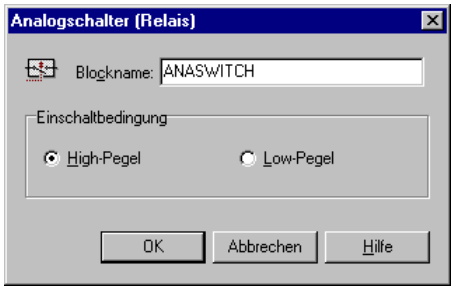
Parameter-dialog:

Analogschalter (Relais)

Typname: ANASWITCH

Funktion: Dieser Funktionsblock bildet einen Analogschalter, d. h. ein Relais, nach: Die Eingangsgröße $x(t)$ wird auf den Ausgang geschaltet, wenn am Steuereingang S High-Pegel bzw. Low-Pegel (umschaltbar) anliegt. Andernfalls wird die Ausgangsgröße auf null gesetzt.

Parameterdialog:

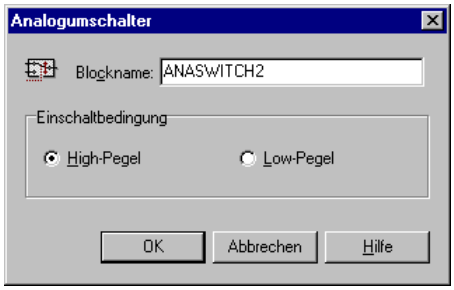


Analogumschalter

Typname: ANASWITCH2

Funktion: Dieser Funktionsblock realisiert einen Analogumschalter. Die Eingangsgröße $x_1(t)$ wird auf den Ausgang geschaltet, wenn am Steuereingang S High-Pegel bzw. Low-Pegel (umschaltbar) anliegt. Andernfalls wird die Ausgangsgröße auf $x_2(t)$ gesetzt.

Parameterdialog:



Digitalbausteine

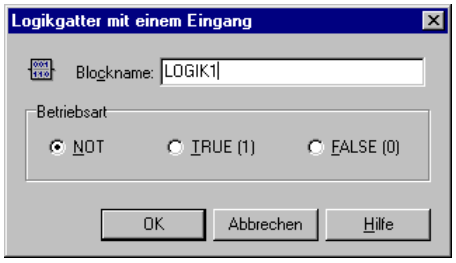
Logikgatter mit einem Eingang

Typname: LOGIK1

Funktion: Dieser Block stellt ein Logikgatter mit einem Eingang dar. Es sind drei verschiedene Betriebsarten möglich:

Negation:	$y = \overline{x}$	(Eingangssignal wird negiert)
TRUE:	$y = 1$	(Ausgang hat immer High-Pegel)
FALSE:	$y = 0$	(Ausgang hat immer Low-Pegel)

Parameterdialog:



Logikgatter mit zwei Eingängen

Typname: LOGIK2

Funktion: Logikgatter mit zwei Eingängen *x* und *y* und dem Ausgang *z*. Es sind folgende Verknüpfungen möglich:

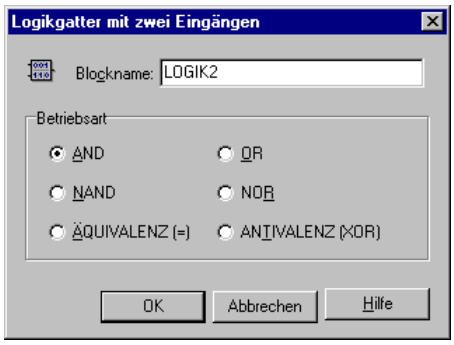
AND:	$z = x \wedge y$
OR:	$z = x \vee y$
NAND:	$z = \overline{x \wedge y}$

NOR: $z = \overline{x \vee y}$

Äquivalenz: $z = x \equiv y$

Antivalenz: $z = x \neq y$

Parameter-dialog:



Logikgatter mit mehreren Eingängen

Typname: LOGIK

Funktion: Logikgatter mit bis zu 50 Eingängen und den Betriebsarten *AND*, *NAND*, *OR* und *NOR*.

Parameter-dialog:

RS-Flip-Flop

Typname: RSFLIPFLOP

Funktion: Dieser Block realisiert ein RS-Flip-Flop. Der Ausgang wird durch den Eingang S gesetzt und durch den Eingang R zurückgesetzt. Das Flip-Flop kann in zwei verschiedenen Betriebsarten betrieben werden:

- In der Betriebsart *statisch* sind die statischen Eingangspegel entscheidend.
- In der Betriebsart *dynamisch* ist das Flip-Flop positiv flankengetriggert, d. h. eine Zustandsänderung findet nur bei Eingangsflanken 0 (Low) → 1 (High) statt.

Nachfolgend ist die Wertetabelle für das RS-Flip-Flop dargestellt.

R	S	y_{neu}
0	0	y_{alt}
0	1	1
1	0	0
1	1	$\bar{y}_{alt}$

Der in der Praxis unbestimmte Zustand R = S = 1 (letzte Zeile der Tabelle) führt zu einem Wechsel des Ausgangszustands bei jedem Simulationsschritt.

Für die Simulation muß der Ausgangspegel des Flip-Flops zu Beginn der Simulation vorgegeben werden.

Parameterdialog:

Ist die Option *Laufzeit nachbilden* aktiviert, so wird der ermittelte Ausgangswert erst einen Simulationsschritt später ausgegeben. Diese Option sollte daher aktiviert werden, wenn mehrere Flip-Flops in Reihe geschaltet werden, z. B. um Schiebe- oder Ringregister nachzubilden oder um z. B. Frequenzteiler aufzubauen (siehe dazu die Beispieldateien SCH_REG.BSY, RING_REG.BSY und TEILER2.BSY im WinFACT-Beispielverzeichnis).

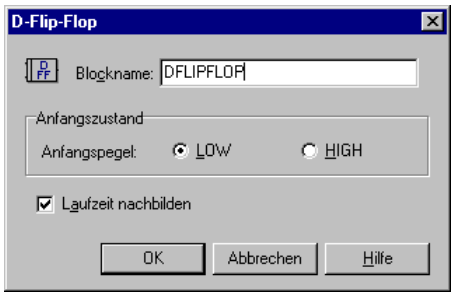
D-Flip-Flop

Typname: DFLIPFLOP

Funktion: Dieser Block stellt ein D-Flip-Flop dar und kann damit als 1-Bit-Speicher oder zur Realisierung von Schieberegistern oder Frequenzteilern verwendet werden. Bei einer positiven Flanke am Takteingang C wird der aktuelle Zustand des Eingangs gespeichert und am Ausgang ausgegeben.

Für die Simulation muß der Ausgangspegel des Flip-Flops zu Beginn der Simulation vorgegeben werden.

Parameterdialog:



Die Einstellung im Feld *Triggerung* ist für diesen Flip-Flop-Typ ohne Bedeutung. Ist die Option *Laufzeit nachbilden* aktiviert, so wird der ermittelte Ausgangswert erst einen Simulationsschritt später ausgegeben. Diese Option sollte daher aktiviert werden, wenn mehrere Flip-Flops in Reihe geschaltet werden, z. B. um Schiebe- oder Ringregister nachzubilden oder um z. B. Frequenzteiler aufzubauen (siehe dazu die Beispieldateien SCH_REG.BSY, RING_REG.BSY und TEILER2.BSY im WinFACT-Beispielverzeichnis).

JK-Flip-Flop

Typname: JKFLIPFLOP

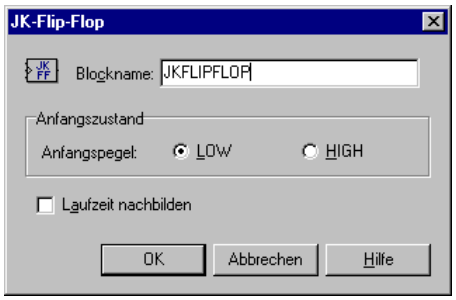
Funktion: Dieser Block realisiert ein positiv-flankengetriggertes JK-Flip-Flop. Bei einer positiven Flanke am Takteingang C wird der Ausgang je nach Zustand des Setzeingangs J und des Rücksetzeingangs K gesetzt bzw. rückgesetzt. Nachfolgend ist die zugehörige Wertetabelle dargestellt.

J	K	y_{neu}
0	0	y_{alt}
0	1	0
1	0	1
1	1	$\bar{y}_{alt}$

Der in der Praxis unbestimmte Zustand J = K = 1 (letzte Zeile der Tabelle) führt zu einem Wechsel des Ausgangszustands bei jedem Simulationsschritt.

Für die Simulation muß der Ausgangspegel des Flip-Flops zu Beginn der Simulation vorgegeben werden.

Parameterdialog:



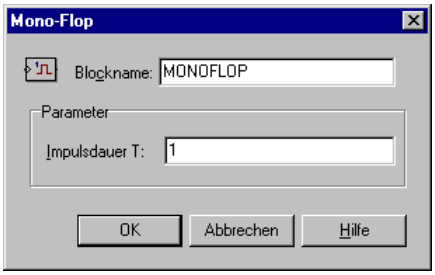
Die Einstellung im Feld *Triggerung* ist für diesen Flip-Flop-Typ ohne Bedeutung. Ist die Option *Laufzeit nachbilden* aktiviert, so wird der ermittelte Ausgangswert erst einen Simulationsschritt später ausgegeben. Diese Option sollte daher aktiviert werden, wenn mehrere Flip-Flops in Reihe geschaltet werden, z. B. um Schiebe- oder Ringregister nachzubilden oder um z. B. Frequenzteiler aufzubauen (siehe dazu die Beispieldateien SCH_REG.BSY, RING_REG.BSY und TEILER2.BSY im WinFACT-Beispielverzeichnis).

Mono-Flop

Typname: MONOFLOP

Funktion: Der Block stellt eine monostabile Kippschaltung (Univibrator) dar. Er erzeugt bei einer positiven Flanke am Eingang am Ausgang einen Impuls vorgebbarer Dauer *T*. Das Mono-Flop ist nicht nachtriggerbar. Trifft ein Impuls am Eingang ein, während der Ausgang noch auf High-Pegel liegt, so wird dieser Impuls ignoriert. Die Impulsdauer sollte ein ganzzahliges Vielfaches der Simulationsschrittweite ΔT betragen.

Parameterdialog:



Parametergrenzen: $T > 0$

Vorwärts-/Rückwärts-Zähler

Typname: ZAEHLER

Funktion: Dieser Systemblock stellt einen positiv-flankengetriggerten Zählerbaustein mit einem vorgebbaren Anfangswert dar. Die Zählrichtung ist intern oder extern umschaltbar. Der Block kann während der Simulation jederzeit durch eine positive Flanke am Reset-Eingang R auf seinen Anfangswert zurückgesetzt werden.

Der Zähler kann auch negative Zählerstände aufweisen.

Parameter-dialog:

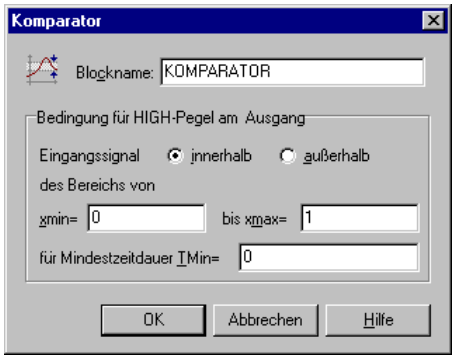
Ist die Option *extern gesteuert* aktiv, so zählt der Baustein bei High-Pegel an Eingang D vorwärts, bei Low-Pegel rückwärts.

Komparator

Typname: KOMPARATOR

Funktion: Dieser Block realisiert einen Fensterkomparator (Fensterdiskriminator). Er liefert am Ausgang High-Pegel, wenn die Eingangsgröße x für eine vorgebbare Mindestzeitdauer T_{min} innerhalb/außerhalb eines bestimmten Wertebereichs $[\,x_{min}\,,\,x_{max}\,]$ liegt. Dadurch ist es möglich, nur kurzzeitig auftretende Bereichsverletzungen zu ignorieren.

Parameter-dialog:



Parameter-grenzen: $x_{min} < x_{max}$, $T_{min} \geq 0$

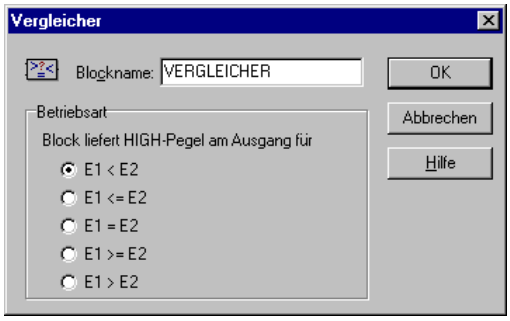
Vergleicher

Typname: VERGLEICHER

Funktion: Der Block vergleicht die beiden analogen Eingangsgrößen E_1 und E_2 und liefert am Ausgang High-Pegel, falls die Vergleichsbedingung erfüllt ist. Es stehen folgende Vergleichsoperatoren zur Auswahl:

$$E_1 < E_2 \quad E_1 \le E_2 \quad E_1 = E_2 \quad E_1 \ge E_2 \quad E_1 > E_2$$

Parameterdialog:

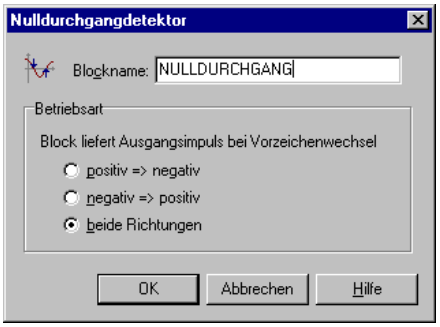


Nulldurchgangdetektor

Typname: NULLDURCHGANG

Funktion: Dieser Block erzeugt einen Ausgangsimpuls, wenn beim Eingangssignal ein Vorzeichenwechsel auftritt. Die Länge des Ausgangsimpulses entspricht der Simulationsschrittweite ΔT. Die Richtung des zu detektierenden Nulldurchgangs kann vorgegeben werden.

Parameter-dialog:



Aktionsblöcke

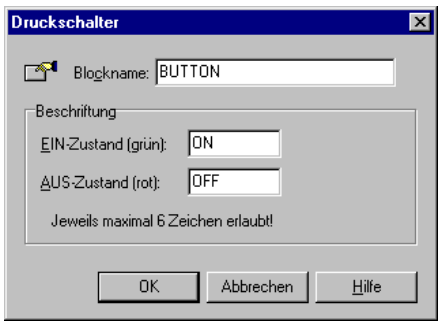
Druckschalter

Typname: BUTTON

Funktion: Dieser Block realisiert einen vom Anwender mit der linken Maustaste zu betätigenden Druckschalter mit Ein-/Ausfunktion und wählbarer Beschriftung.

Steuerfenster des Schalters im EIN- bzw. AUS-Zustand

Parameter-dialog:



Schieberegler (Potentiometer)

Typname: POTI

Funktion: Dieser Block realisiert einen vom Anwender mit der linken Maustaste zu betätigenden Schieberegler. Der Baustein kann wahlweise mit (Voreinstellung) oder ohne Eingang betrieben werden. In der Betriebsart mit Eingang wird der anliegende Eingangswert mit der *Verstärkung* des Schiebereglers multipliziert ausgegeben. In der Betriebsart ohne Eingang liegt der Ausgangswert je nach Potistellung zwischen 0 und der vorgegebenen Verstärkung. Zusätzlich kann eine Digitalanzeige der auf die Potistellung bezogenen Verstärkung als Absolut- oder Realtivwert (in %) erfolgen.

Steuerfenster des Schiebereglers mit absoluter bzw. relativer Digitalanzeige

Parameterdialog:

Drehregler

Typname: DREHREGLER

Funktion: Dieser Block realisiert einen vom Anwender mit der linken Maustaste zu betätigenden Drehregler. Der Baustein kann wahlweise mit oder ohne Eingang betrieben werden. In der Betriebsart *mit Eingang* wird der anliegende Eingangswert mit der aktuellen Einstellung des Drehreglers multipliziert ausgegeben. In der Betriebsart *ohne Eingang* wird der aktuell eingestellte Skalenwert direkt ausgegeben.

Steuerfenster des Drehreglers

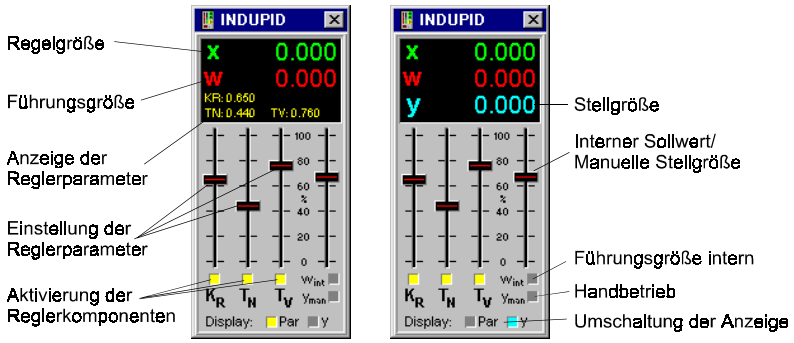
Parameter-dialog:

Industrie-PID-Regler

Typname: INDUPID

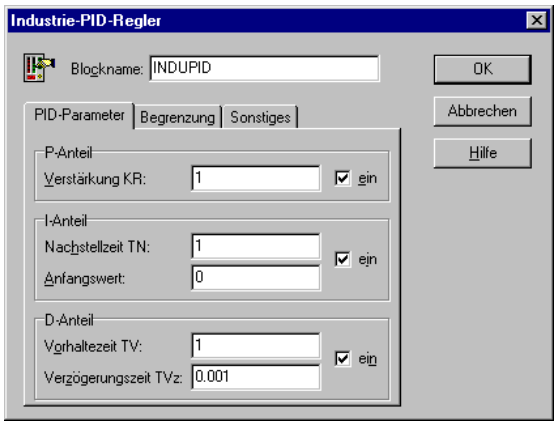
Funktion: Dieser Block realisiert einen PID-Regler mit einer Steuer- und Visualisierungsoberfläche, die einem Industrieregler nachempfunden wurde. Von der Funktionalität her entspricht dieser Blocktyp dem Standard-PID-Regler von BORIS, weist aber Soll- und Istwert (Führungsgröße *w* bzw. Regelgröße *x*) als getrennte Eingänge auf. Außerdem kann der Sollwert auf Wunsch auch intern generiert werden; der erste Eingang des Blocks ist dann ohne Bedeutung. Bei Bedarf kann der Regler auch im Handbetrieb (manuelle Vorgabe der Stellgröße) gefahren werden.

Die Reglerparameter sowie der interne Sollwert (falls aktiviert) können über Schieberegler variiert werden. Die einzelnen Regleranteile (P-, I- und D-Anteil) sind über Mausklick zu- bzw. abschaltbar. Im Display des Reglers werden neben Soll- und Istwert auch die aktuellen Reglerparameter bzw. wahlweise die aktuelle Stellgröße *y* angezeigt.



Steuer- und Visualisierungsfenster des Industrie-PID-Reglers mit Anzeige der Reglerparameter (links) bzw. der Stellgröße (rechts)

Parameterdialog:



Kommunikation

DDE-Eingang

Typname: DDEIN

Funktion: Dieser Block ermöglicht das Einlesen von Werten aus anderen Windows-Anwendungen (z. B. EXCEL oder LabView) über *Dynamic Data Exchange* (DDE).

Der Datenlieferant (*Server*) wird über folgende Parameter spezifiziert:

Name Name der Server-Applikation (z. B. EXCEL)

Thema Thema der DDE-Konversation (z. B. Name der EXCEL-Tabelle, also z. B. TAB1)

Item Spezifizierung des einzulesenden Wertes (z. B. Z1S1 für die erste Zeile, erste Spalte der EXCEL-Tabelle)

Bei der Spezifizierung des Items kann auch eine Zählvariable der Form *{#n}* eingefügt werden, um z. B. alle Werte fortlaufend aus einer Spalte einer EXCEL-Tabelle zu lesen. Soll beispielsweise die erste Spalte gelesen werden, wobei der erste Wert aus der ersten Zeile, der zweite aus der zweiten Zeile usw. gelesen wird, so lautet der Eintrag *Z{#1}S1*. Soll erst in der zweiten Zeile begonnen werden, lautet der Eintrag dementsprechend *Z{#2}S1*. Sollen die Werte hingegen z. B. zeilenweise aus der ersten Zeile gelesen werden, lautet der Eintrag *Z1S{#1}* usw. Man beachte hierbei, daß der erste Wert grundsätzlich bei der Initialisierung der Simulation gelesen wird, d. h. im ersten Simulationsschritt bereits der zweite Wert gelesen wird usw.

Parameter-dialog:

Über die Schaltfläche *Verknüpfung einfügen* kann eine zuvor von einer anderen Anwendung in die Zwischenablage kopierte Verknüpfung automatisch eingefügt werden (in obigem Dialog z. B. ein Verweis auf eine Zelle eines Excel-Arbeitsblattes).

Hinweis: Im Lieferumfang von WinFACT 98 befindet sich ein Demo-Programm mit Namen DDEDEMO.EXE, mit dem Sie die DDE-Fähigkeiten von BORIS ausprobieren können. Starten Sie das Programm, starten Sie dann BORIS und laden Sie dort die Beispieldatei DDEDEMO.BSY. Danach starten Sie die Simulation. Sie können nun in DDEDEMO Daten editieren, die dann in BORIS angezeigt werden; umgekehrt werden die von BORIS generierten Daten in DDEDEMO angezeigt!

DDE-Ausgang

Typname: DDEOUT

Funktion: Dieser Block ermöglicht die Ausgabe von Werten zu anderen Windows-Anwendungen (z. B. EXCEL oder LabView) über *Dynamic Data Exchange* (DDE).

Der Datenempfänger (*Server*) wird über folgende Parameter spezifiziert:

Name Name der Server-Applikation (z. B. EXCEL)

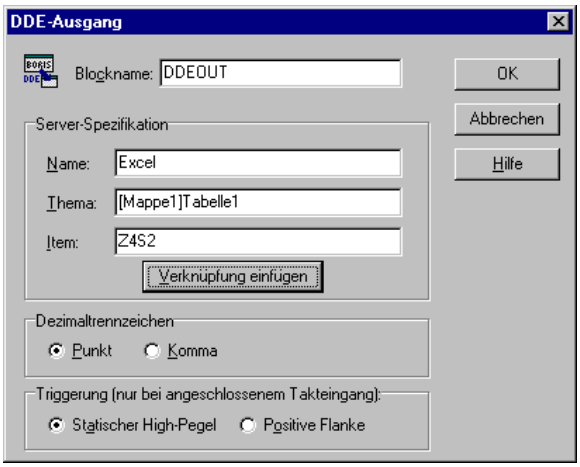
Thema Thema der DDE-Konversation (z. B. Name der EXCEL-Tabelle, also z. B. TAB1)

Item Spezifizierung des Ziels für den ausgegebenen Wert (z. B. Z1S1 für die erste Zeile, erste Spalte der EXCEL-Tabelle)

Bei der Spezifizierung des Items kann auch eine Zählvariable der Form *{#n}* eingefügt werden, um z. B. alle Werte fortlaufend in eine Spalte einer EXCEL-Tabelle zu schreiben. Soll beispielsweise die erste Spalte beschrieben werden, wobei der erste Wert in die erste Zeile, der zweite in die zweite Zeile usw. geschrieben wird, so lautet der Eintrag *Z{#1}S1*. Soll erst in der zweiten Zeile begonnen werden, lautet der Eintrag dementsprechend *Z{#2}S1*. Sollen die Werte hingegen z. B. zeilenweise in die erste Zeile geschrieben werden, lautet der Eintrag *Z1S{#1}* usw. Man beachte hierbei, daß der erste Wert grundsätzlich bei der Initialisierung der Simulation geschrieben wird, d. h. im ersten Simulationsschritt bereits der zweite Wert geschrieben wird usw.

Bei angeschlossenem Triggereingang C erfolgt eine Ausgabe wahlweise nur bei einer positiven Flanke am Triggereingang oder bei statischem High-Pegel. Das Dezimaltrennzeichen (Punkt bzw. Komma) kann über die entsprechenden Schalter frei gewählt werden.

Parameter-dialog:



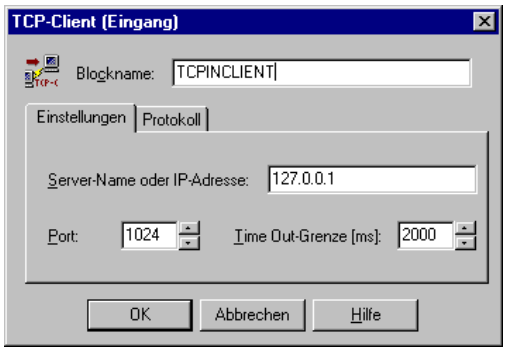
Über die Schaltfläche *Verknüpfung einfügen* kann eine zuvor von einer anderen Anwendung in die Zwischenablage kopierte Verknüpfung automatisch eingefügt werden (in obigem Dialog z. B. ein Verweis auf eine Zelle eines Excel-Arbeitsblattes).

TCP-Client (Eingang)

Typname: TCPINCLIENT

Funktion: Dieser Eingangsblock realisiert einen Client, der Daten nach dem TCP/IP-Protokoll empfängt. Der aktuelle Status der Verbindung wird über ein separates Statusfenster angezeigt. Die genaue Funktionalität erläutert die Datei TCPDEMO.BSY bzw. die Dateien TCPCLIENT.BSY/TCPSERVER.BSY (beide Dateien in getrennte BORIS-Instanzen laden und dann beide Simulationen starten!) aus dem Examples-Verzeichnis.

Parameterdialog:

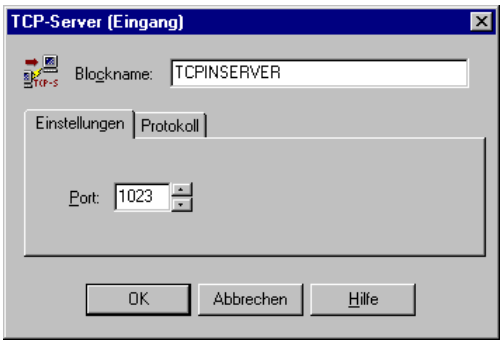


TCP-Server (Eingang)

Typname: TCPINSERVER

Funktion: Dieser Eingangsblock realisiert einen Server, der Daten nach dem TCP/IP-Protokoll empfängt. Der aktuelle Status der Verbindung wird über ein separates Statusfenster angezeigt. Die genaue Funktionalität erläutert die Datei TCPDEMO.BSY bzw. die Dateien TCPCLIENT.BSY/TCPSERVER.BSY (beide Dateien in getrennte BORIS-Instanzen laden und dann beide Simulationen starten!) aus dem Examples-Verzeichnis.

Parameter-dialog:

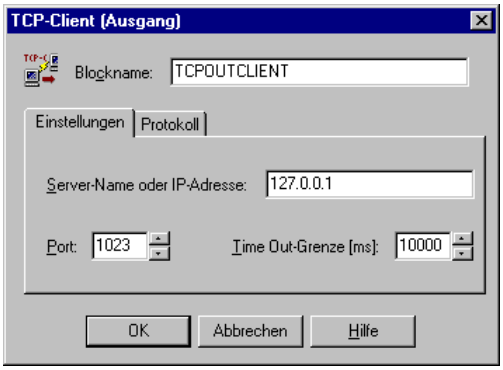


TCP-Client (Ausgang)

Typname: TCPOUTCLIENT

Funktion: Dieser Ausgangsblock realisiert einen Client, der Daten nach dem TCP/IP-Protokoll versendet. Der aktuelle Status der Verbindung wird über ein separates Statusfenster angezeigt. Die genaue Funktionalität erläutert die Datei TCPDEMO.BSY bzw. die Dateien TCPCLIENT.BSY/TCPSERVER.BSY (beide Dateien in getrennte BORIS-Instanzen laden und dann beide Simulationen starten!) aus dem Examples-Verzeichnis.

Parameter-dialog:



TCP-Server (Ausgang)

Typname: TCPOUTSERVER

Funktion: Dieser Ausgangsblock realisiert einen Server, der Daten nach dem TCP/IP-Protokoll versendet. Der aktuelle Status der Verbindung wird über ein separates Statusfenster angezeigt. Die genaue Funktionalität erläutert die Datei TCPDEMO.BSY bzw. die Dateien TCPCLIENT.BSY/TCPSERVER.BSY (beide Dateien in getrennte BORIS-Instanzen laden und dann beide Simulationen starten!) aus dem Examples-Verzeichnis.

Parameterdialog:

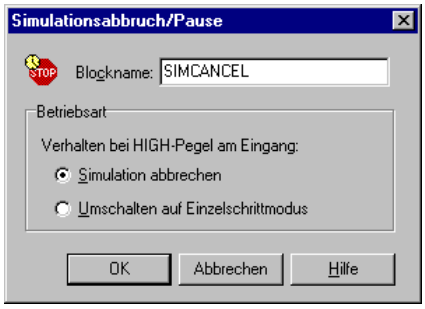
Simulationssteuerung

Simulationsabbruch/Pause

Typname: SIMCANCEL

Funktion: Dieser Block bricht die Simulation ab bzw. schaltet in den Einzelschrittmodus, sobald an seinem Eingang High-Pegel anliegt.

Parameter-dialog:



Simulationsverzögerung

Typname: SIMDELAY

Funktion: Dieser Block verzögert die Simulation bei jedem Simulationsschritt um eine vorgebbare Zeit in Millisekunden. Er kann benutzt werden, um auf schnellen Rechnern eine künstliche Verlangsamung des Simulationsablaufs zu erreichen.

Parameter-dialog:

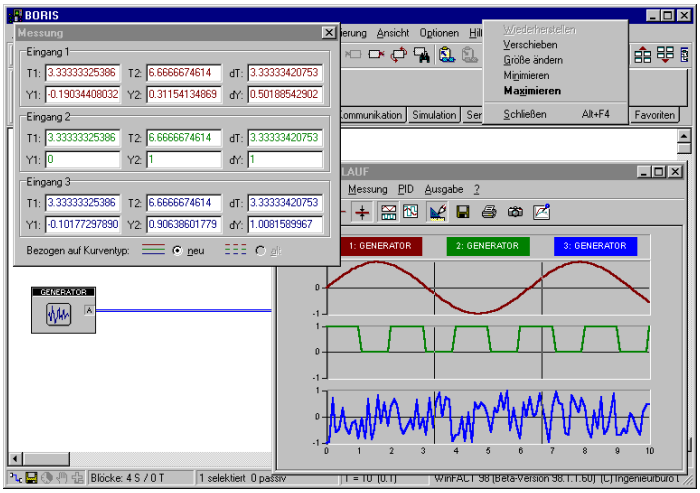
Ausgangsblöcke (Senken)

Zeitverlauf

Typname: ZEITVERLAUF

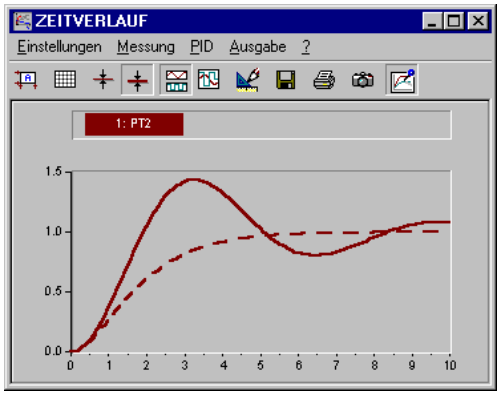
Funktion: Dieser Ausgangsblock ermöglicht die gleichzeitige Darstellung von bis zu drei Zeitverläufen in einem beliebig vergrößerbaren Anzeigefenster. Die Kurven können in ein gemeinsames oder in getrennte Diagramme, mit und ohne Raster gezeichnet werden. Die Skalierung der Koordinatenachsen kann manuell oder automatisch erfolgen. Die Amplitudenwerte können sowohl linear als auch logarithmisch aufgetragen werden. Außerdem ist ein direktes Abspeichern von Kurven in SIM-Dateien möglich.

Über eine komfortable Meßfunktion können den Diagrammen auf einfache Weise Meßwerte entnommen werden. Da die Zeitverläufe der vorangegangenen Simulation jeweils "eingefroren" werden können, sind auch Trends bei Parametervariationen - z. B. von Reglerparametern - darstellbar. Die Meßfunktion kann bei Bedarf auch auf die eingefrorenen Kurven angewendet werden. Speziell für regelungstechnische Anwendungen (z. B. das Ablesen von Ausregelzeiten) kann zusätzlich ein *Toleranzband* mit frei wählbarer Breite in die Diagramme eingeblendet werden.



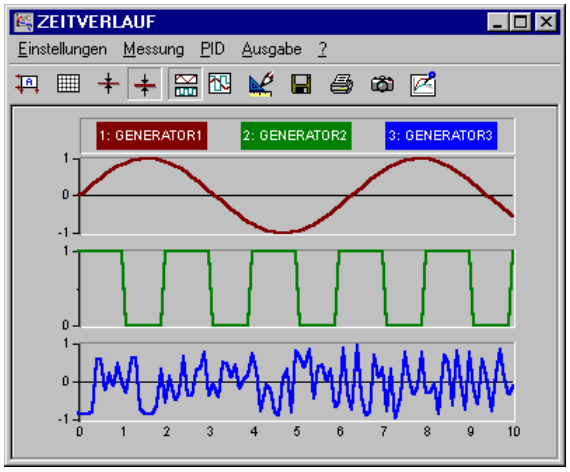
Bildschirm bei aktivierter Meßfunktion

Weiterhin ist aus dem Anzeigefenster heraus der direkte Entwurf von PID-Reglern nach Einstellregeln möglich. Auf diese Funktion wird an anderer Stelle im Kapitel *Entwurf von PID-Reglern* detailliert eingegangen.



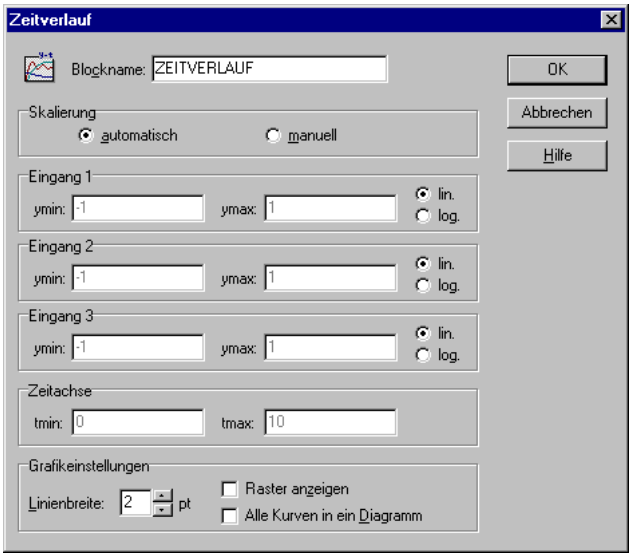
Zeitverlauffenster bei aktivierter Speicherfunktion: Die im vorangegangenen Simulationslauf ermittelte Kurve wird gestrichelt eingezeichnet

Anzeige-fenster:



Oberhalb der Diagramme werden die Bezeichner der zugehörigen Systemblöcke (hier *Generator1*, *Generator2* und *Generator3*) angegeben. Die Toolbar unterhalb des Fenstermenüs erlaubt den Direktzugriff auf die wichtigsten Menüoptionen bzw. Einstellungen des Parameterdialogs, der im übrigen auch über die Menüoption EINSTELLUNGEN erreicht werden kann.

Parameter-dialog:



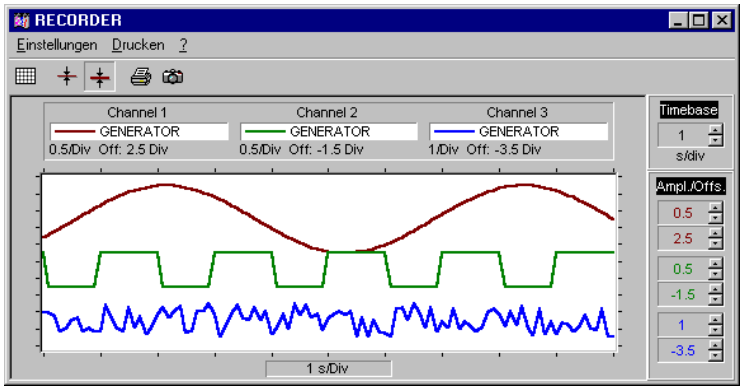
Parameter-grenzen: Für jede Kurve stehen intern maximal 32000 Punkte zur Verfügung. Ist die Anzahl der Simulationsschritte größer, so werden Zwischenwerte übersprungen ("komprimierte" Darstellung). Beträgt die Anzahl der Simulationsschritte beispielsweise 64000, so wird nur jeder zweite Simulationsschritt grafisch dargestellt. Dies ist zu beachten, wenn etwa sehr kurze Impulse dargestellt werden sollen. Diese komprimierte Darstellung wird daher durch den Schriftzug *COMPR!* in rot auf gelbem Grund in der linken oberen Ecke des Anzeigefensters angezeigt.

y-t-Schreiber (Recorder)

Typname: RECORDER

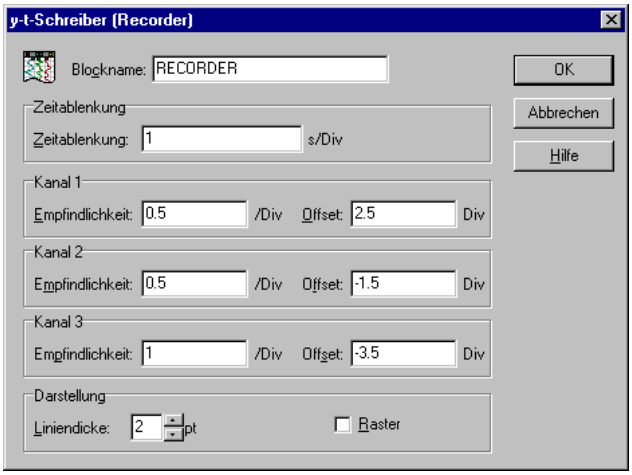
Funktion: Dieser Block bildet einen y-t-Schreiber nach und ist damit insbesondere für Langzeitaufzeichnungen geeignet. Der Block kann bis zu drei Eingangskanäle verarbeiten. Zeitbasis sowie Ablenkempfindlichkeit und Offset der einzelnen Kanäle sind frei wählbar.

Anzeige-fenster:



Oberhalb der Diagramme werden die Bezeichner der zugehörigen Systemblöcke (hier *Generator1*, *Generator2* und *Generator3*) angegeben. Die Toolbar unterhalb des Fenstermenüs erlaubt den Direktzugriff auf die wichtigsten Menüoptionen bzw. Einstellungen des Parameterdialogs, der im übrigen auch über die Menüoption EINSTELLUNGEN erreicht werden kann.

Parameter-dialog:

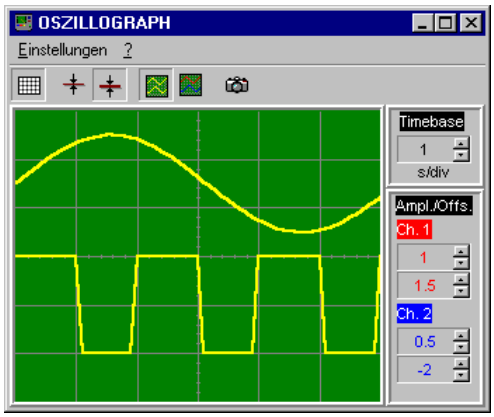


Oszillograph

Typname: OSZILLOGRAPH

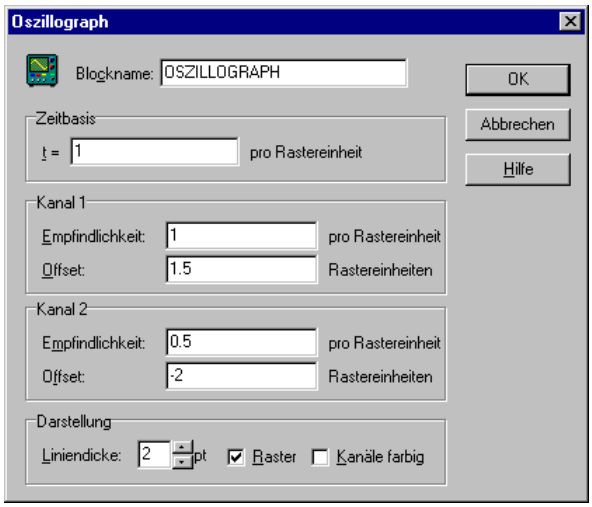
Funktion: Dieser Block bildet einen Zweikanal-Oszillographen auf dem Bildschirm nach, der beliebig verkleinert und vergrößert werden kann. Die Ablenkempfindlichkeit und der Nullpunkt sind für beide Kanäle getrennt einstellbar. Ebenso ist die Zeitablenkung variierbar. Beide Kanäle können auf Wunsch mehrfarbig dargestellt werden.

Anzeige-fenster:



Die eingestellten Empfindlichkeiten werden in der rechten oberen Ecke angegeben. Die Toolbar unterhalb des Fenstermenüs erlaubt den Direktzugriff auf die wichtigsten Einstellungen des Parameterdialogs, der im übrigen auch über die Menüoption EINSTELLUNGEN erreicht werden kann.

Parameter-dialog:



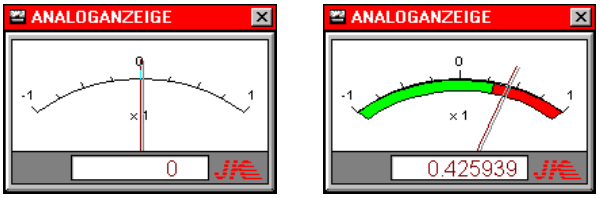
Parameter-grenzen: Für jede Kurve stehen intern maximal 32000 Punkte zur Verfügung. Ist die Zahl der Simulationsschritte, die auf die Breite des Oszillographs entfallen, größer, so werden Zwischenwerte übersprungen ("komprimierte" Darstellung). Beträgt die Anzahl der Schritte beispielsweise 64000, so wird nur jeder zweite Simulationsschritt grafisch dargestellt. Dies ist zu beachten, wenn etwa sehr kurze Impulse dargestellt werden sollen. Diese komprimierte Darstellung wird daher durch den Schriftzug *COMPR!* in rot auf gelbem Grund in der rechten unteren Ecke des Anzeigefensters angezeigt (vgl. Zeitverlauf-Block!)

Analoganzeige

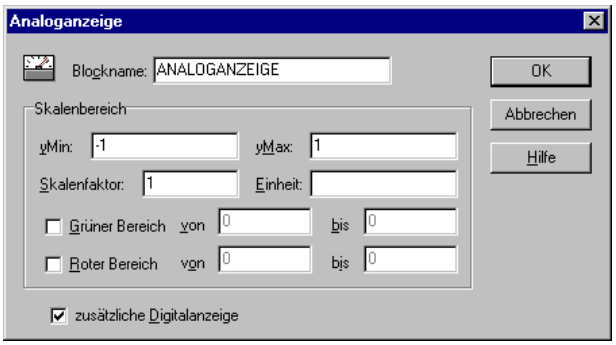
Typname: ANALOGANZEIGE

Funktion: Dieser Block stellt die Nachbildung eines Analoginstruments - auf Wunsch mit Digitalanzeige - dar. Bestimmte Skalenbereiche können rot oder grün dargestellt und die Anzeige mit einer Einheit (z. B. "V") versehen werden.

Anzeige-fenster:



Parameter-dialog:



Für den Skalenbereich $y_{\min}$, $y_{\max}$ sollten möglichst ganzzahlige Werte vorgegeben werden.

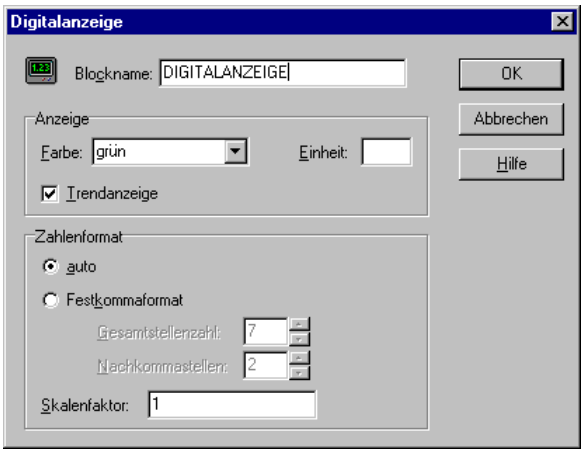
Digitalanzeige

Typname: DIGITALANZEIGE

Funktion: Dieser Block bildet ein Digitalinstrument nach. Das zugehörige Anzeigefenster besitzt eine feste Größe. Auf Wunsch kann ein zusätzlicher Skalenfaktor und eine Einheit (z. B. "V") sowie eine Trendanzeige (Eingangswert steigend/fallend) vorgegeben werden. Die Ausgabe des numerischen Werts erfolgt wahlweise mit möglichst geringer oder fest vorgegebener Stellenzahl.

Anzeige-fenster:

Parameter-dialog:



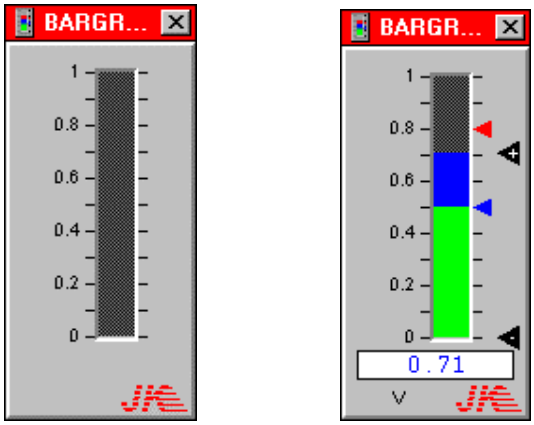
Balkendiagramm

Typname: BARGRAPH

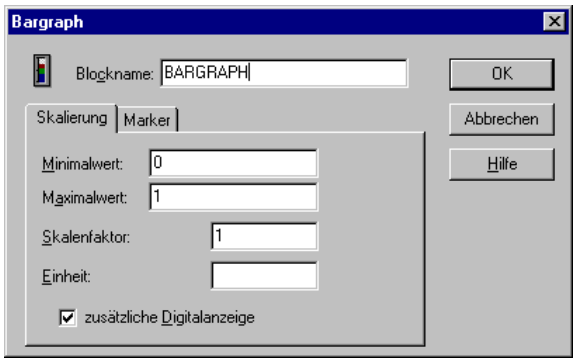
Funktion: Dieser Ausgabeblock stellt ein Balkendiagramm (Bargraph) dar, das in verschiedenen Modi betrieben werden kann. Im einzelnen sind folgende Optionen möglich:

- Setzen von ein oder zwei Markern, bei deren Überschreitung die Anzeige ihre Farbe wechselt.
- Setzen von Markern für minimal und maximal erreichten Wert.
- Zusätzliche Digitalanzeige, Skalenfaktor und Einheit (z. B. "V")

Anzeigefenster:



Parameterdialog:

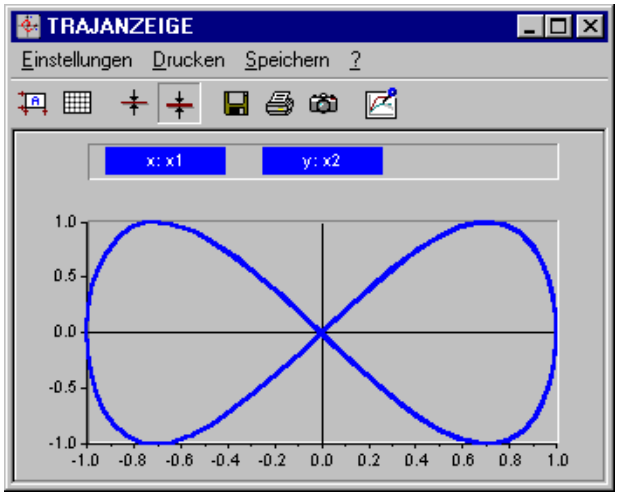


Trajektorienanzeige

Typname: TRAJANZEIGE

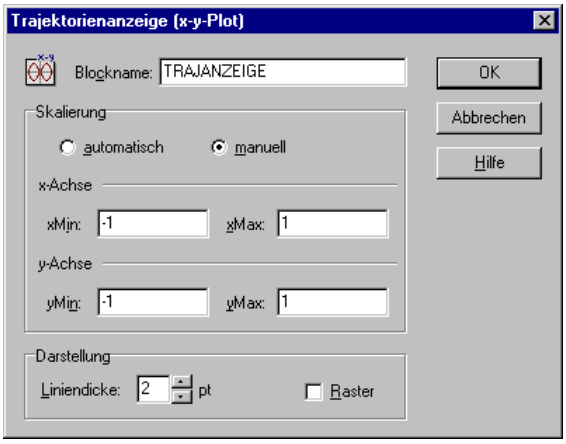
Funktion: Die Trajektorienanzeige erlaubt die Darstellung zweier Eingangsgrößen $x(t)$ und $y(t)$ in der x-y-Ebene mit der Zeit t als Kurvenparameter. Das zugehörige Anzeigefenster kann beliebig verkleinert und vergrößert werden. Die Skalierung der Koordinatenachsen kann manuell oder automatisch, mit oder ohne Raster erfolgen. Wie beim Zeitverlauf-Block steht auch hier eine Speicherfunktion für die vorangegangene Simulation zur Verfügung.

Anzeigefenster:



Oberhalb des Diagramms werden die Bezeichner der zugehörigen Systemblöcke (hier *x1* und *x2*) angegeben. Die Toolbar unterhalb des Fenstermenüs erlaubt den Direktzugriff auf die wichtigsten Einstellungen des Parameterdialogs, der im übrigen auch über die Menüoption EINSTELLUNGEN erreicht werden kann.

Parameter-dialog:



Parameter-grenzen: Für jede Kurve stehen intern maximal 32000 Punkte zur Verfügung. Ist die Anzahl der Simulationsschritte größer, so werden Zwischenwerte übersprungen ("komprimierte" Darstellung). Beträgt die Anzahl der Simulationsschritte beispielsweise 64000, so wird nur jeder zweite Simulationsschritt grafisch dargestellt. Dies ist zu beachten, wenn etwa sehr kurze Impulse dargestellt werden sollen. Diese komprimierte Darstellung wird daher durch den Schriftzug *COMPR!* in rot auf gelbem Grund in der rechten unteren Ecke des Anzeigefensters angezeigt (vgl. Zeitverlauf-Block!).

Statusanzeige

Typname: STATUSANZEIGE

Funktion: Dieser Block erzeugt eine visuelle und auf Wunsch eine zusätzliche akustische Kontrollausgabe beim Über- bzw. Unterschreiten eines vorgebbaren Schwellwerts. Der Text im Display ist für den EIN- und AUS-Zustand getrennt vorgebbar.

Anzeige-fenster:

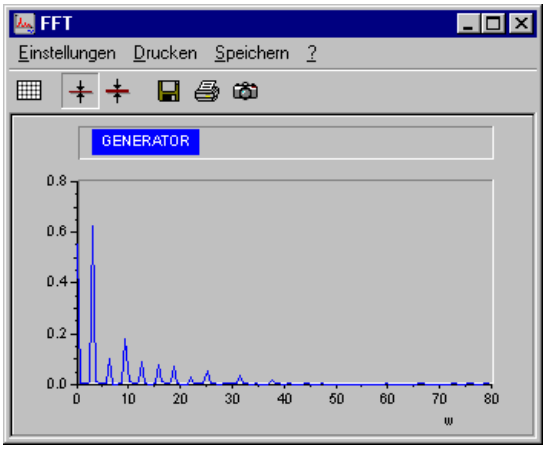
Parameter-dialog:

Fast-Fourier-Transformation (FFT)

Typname: FFT

Funktion: Dieser Block ermöglicht die Berechnung des Amplitudenspektrums des Eingangssignals über die Fast-Fourier-Transformation (FFT). Die Stützstellenzahl und das Zeitfenster für die Transformation sind wählbar.

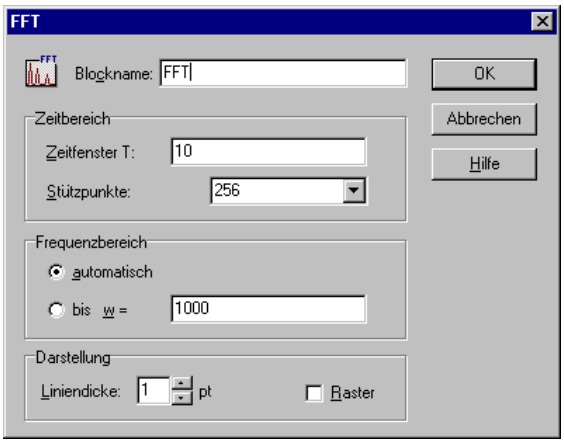
Anzeige-fenster:



Oberhalb des Diagramms wird der Bezeichner des zugehörigen Systemblocks (hier *Generator*) angegeben. Die Toolbar unterhalb des Fenstermenüs erlaubt

den Direktzugriff auf die wichtigsten Einstellungen des Parameterdialogs, der im übrigen auch über die Menüoption EINSTELLUNGEN erreicht werden kann.

Parameter-dialog:



Die einzelnen Parameter haben folgende Bedeutung:

- Das *Zeitfenster T* gibt die Länge des Zeitfensters an, innerhalb dessen die Stützstellen für die Transformation liegen. Hierfür sollte in der Regel die Simulationsdauer gewählt werden; die Transformation erfolgt dann nach Beendigung der Simulation. Die Länge des Zeitfensters darf nicht größer als die Simulationsdauer gewählt werden, da ansonsten keine Transformation und damit auch keine Anzeige erfolgt.
- Das Zeitfenster legt die kleinste Frequenz $\omega_{\min}$ fest, für die das Spektrum berechnet wird. Es gilt

$$\omega_{\min} = \frac{2\pi}{T}.$$

- Die Anzahl der *Stützpunkte* ist immer eine Zweierpotenz und legt die obere Frequenz des berechneten Spektrums fest. Bei einer Zahl von *n* Stützstellen ergibt sich die obere Frequenz zu

$$\omega_{\max} = \left(\frac{n}{2} - 1\right)\omega_{\min}.$$

- Die Skalierung der Frequenzachse wird über die Einstellungen im Gruppenfeld *Frequenzachse* beeinflußt. Bei der automatischen Skalierung wird der gesamte Bereich bis $\omega_{\max}$ dargestellt.
- Über das Gruppenfeld *Anzeige* kann die Liniendicke der Kurve eingestellt sowie ein Koordinatenraster zugeschaltet werden.

ISM-Modul (Ausgang) *

Typname: ISMOUT

Funktion: Ermöglicht die Ausgabe von Signalen über das ISM-110-Meßmodul der Fa. Gantner. Hinweise dazu entnehmen Sie bitte der zugehörigen Dokumentation.

C-Code (Ausgang) *

Typname: CCODEOUTPUT

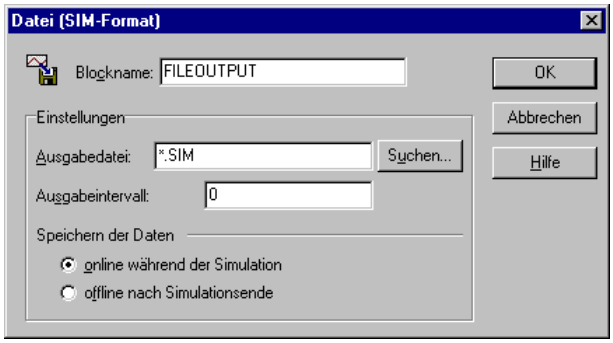
Funktion: Liefert die Schnittstelle für einen Hardware-Ausgang in Verbindung mit dem BORIS-C-Code-Generator. Einzelheiten dazu entnehmen Sie bitte der zugehörigen Dokumentation.

Datei (File-Output)

Typname: FILEOUTPUT

Funktion: Ermöglicht die Ausgabe eines Signals $u(t)$ in Form von Wertepaaren (t_i, u_i) in eine Datei vom Typ SIM.

Parameterdialog:



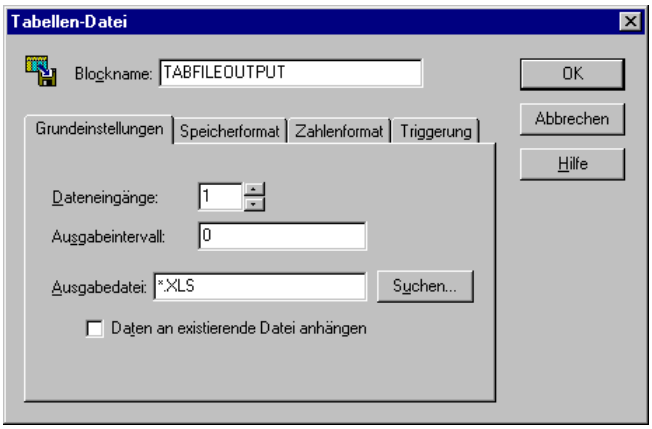
Wird für *Ausgabedatei* keine Extension angegeben, wird die Extension SIM benutzt. Über die Schaltfläche *Suchen* kann ein Dateieingabedialog angefordert werden. Der Parameter *Ausgabeintervall* gibt an, in welchem zeitlichen Abstand die einzelnen Wertepaare abgespeichert werden. Wird z. B. bei einer Simulationsschrittweite von 0.1 ein Ausgabeintervall von 0.5 gewählt, so wird lediglich jedes fünfte Wertepaar abgespeichert. Wird das Ausgabeintervall zu null gewählt (Voreinstellung), so wird jeder simulierte Wert auch abgespeichert. Das Speichern der Daten kann *online* während der Simulation oder *offline* nach Simulationsende erfolgen. Letztere Einstellung empfiehlt sich insbesondere bei Echtzeitsimulationen, da in diesem Fall während der Simulation kein zeitaufwendiger Zugriff auf die Festplatte erfolgen muß. Bei Offline-Speicherung werden die Daten im Arbeitsspeicher zwischengelagert.

Tabellen-Datei (Excel-Format)

Typname: TABFILEOUTPUT

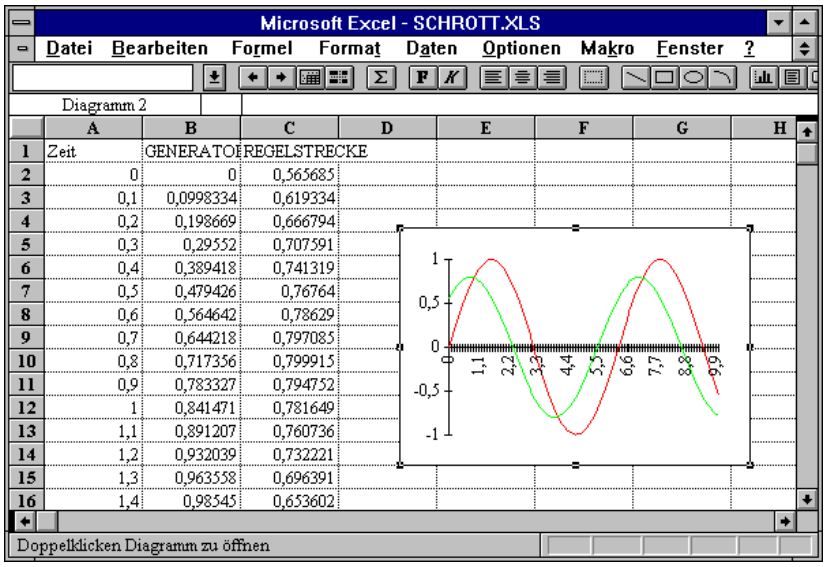
Funktion: Dieser Block ermöglicht das Abspeichern von bis zu 49 Zeitverläufen in Tabellenform. Die abgespeicherten Dateien können dann auf einfache Weise direkt mit Tabellenkalkulationen wie z. B. Excel weiterverarbeitet werden. Die Dateien besitzen die Extension XLS. Die Abspeicherung der Daten kann periodisch oder extern getriggert erfolgen: Ist der Triggereingang C offen, erfolgt ein periodisches Abspeichern, ist er belegt, erfolgt ein Abspeichern wahlweise bei einer positiven Flanke am Triggereingang oder bei statischem High-Pegel.

Parameterdialog:



Die Parameter haben folgende Bedeutung:

- *Dateneingänge* gibt die Anzahl der Blockeingänge (1 bis 49) an.
- *Ausgabedatei* gibt den Namen der Datei an, in der die Daten abgelegt werden. Über *Suchen* kann ein Dateieingabedialog angefordert werden. Die Zeitverläufe werden in dem Zeitabstand abgespeichert, der unter *Ausgabeintervall* angegeben ist. Beträgt dieser Wert 0, so werden die Werte bei jedem Simulationsschritt abgespeichert.
- Ist die Option *mit Spaltenüberschrift* aktiviert, so werden in der ersten Zeile der Datei die Bezeichner der den jeweiligen Eingangsgrößen entsprechenden Blöcke ausgegeben.
- Ist die Option *mit Zeitwerten* aktiviert, so enthält die erste Spalte der abgelegten Tabelle jeweils den aktuellen Zeitwert, andernfalls enthält sie die erste Zustandsgröße und eine Zeitparametrierung findet nicht statt.
- Über *Spaltentrennzeichen* wird angegeben, wie die einzelnen Spalten der Tabelle voneinander getrennt werden. Die Einstellungen unter *Dezimaltrennzeichen* geben an, welches Zeichen für die Darstellung des Dezimalpunkts gewählt wird.



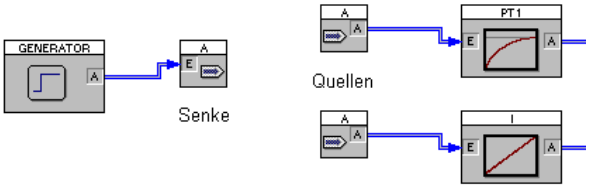
	A	B	C
1	Zeit	GENERATOR	REGELSTRECKE
2	0	0	0,565685
3	0,1	0,0998334	0,619334
4	0,2	0,198669	0,666794
5	0,3	0,29552	0,707591
6	0,4	0,389418	0,741319
7	0,5	0,479426	0,76764
8	0,6	0,564642	0,78629
9	0,7	0,644218	0,797085
10	0,8	0,717356	0,799915
11	0,9	0,783327	0,794752
12	1	0,841471	0,781649
13	1,1	0,891207	0,760736
14	1,2	0,932039	0,732221
15	1,3	0,963558	0,696391
16	1,4	0,98545	0,653602

Weiterverarbeitung von Dateien mit Excel

Signalsenke

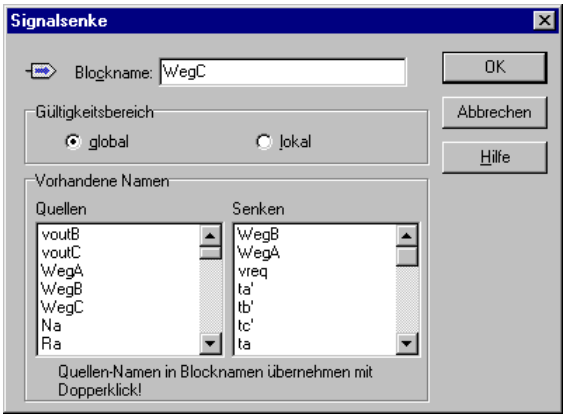
Typname: SENKE

Funktion: Dieser Block dient zusammen mit dem Eingangsblocktyp *Signalquelle* zur Realisierung "drahtloser" Verbindungen zwischen Blöcken. Dabei "versendet" eine Signalsenke ihr Eingangssignal unter einem bestimmten Namen (dem Namen des Blocks). Dieses Signal kann dann an beliebigen - auch mehreren - Stellen in der Systemstruktur von einer Signalquelle mit gleichem Namen wieder empfangen werden. Groß- und Kleinschreibung der Blocknamen wird dabei nicht unterschieden. Signalsenken können *lokale* oder *globale* Gültigkeit haben. Während sie im ersten Fall nur innerhalb der zugehörigen System- bzw. Superblockdatei bekannt sind, gelten sie im globalen Fall auch über die Dateigrenzen hinaus. Der Einsatz beider Blocktypen empfiehlt sich insbesondere bei komplexen, stark vermaschten Systemstrukturen, um die Anzahl der sichtbaren Verbindungen gering zu halten.



Das Quellen/Senken-Konzept

Parameterdialog:



In der linken Listbox werden die Namen der bereits vorhandenen Quellen in alphabetischer Reihenfolge angezeigt, in der rechten Listbox die der Senken. Ein Quellen-Name kann durch Doppelklick direkt als Senken-Name übernommen werden.

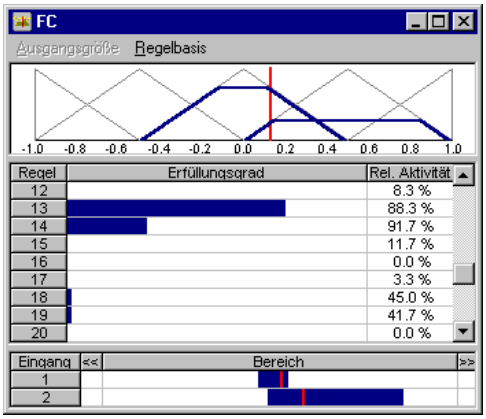
Sonstige Systemblöcke

Fuzzy Controller

Typname: FC

Funktion: Dieser Systemblock realisiert einen Fuzzy Controller, der über eine FUZ-Datei parametriert wird, die mit Hilfe der WinFACT-Fuzzy-Shell FLOP erstellt werden kann. Der Fuzzy Controller-Block besitzt ein als *Fuzzy-Debugger* bezeichnetes Kontrollfenster, das während der Simulation aus seiner Symboldarstellung geholt werden kann und wichtige Hinweise auf die innere Funktion des Controllers gibt. Der Fuzzy-Debugger bietet damit eine wesentliche Unterstützung beim interaktiven Fuzzy Controller-Entwurf. Die nachfolgenden Grafiken zeigen den Aufbau des Debuggers sowie seinen Parameterdialog. Der Fuzzy-Debugger zeigt online während der Simulation folgende Größen an:

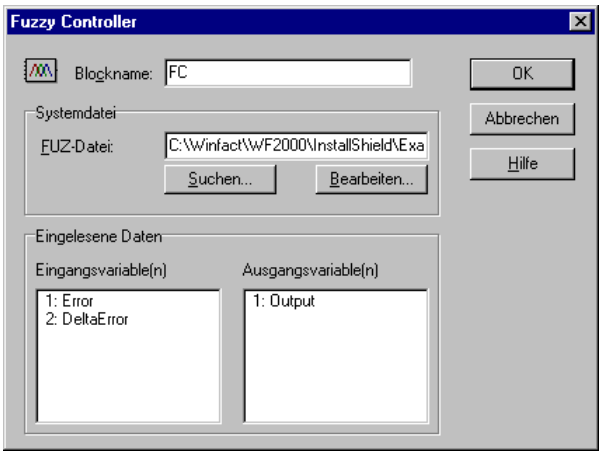
- Im oberen Fensterdrittel die Fuzzy-Mengen der gewählten FC-Ausgangsgröße (grau), die aktuelle scharfe Ausgangsgröße (rot) sowie die resultierende Ausgangs-Fuzzy-Menge (blau).
- Im mittleren Fensterdrittel die aktuellen Erfüllungsgrade aller Regeln (blaue Balken) sowie ihre relative Aktivität. Besitzt eine Regel z. B. eine relative Regelaktivität von 30%, so bedeutet dies, daß die Regel in 30% aller Simulationsschritte einen Erfüllungsgrad größer null aufgewiesen hat.
- Im unteren Fensterdrittel die aktuellen scharfen Eingangswerte (rot) sowie die bisherige Ausnutzung des Eingangsgrößenbereichs des Fuzzy Controllers (blau). Bei Unter- bzw. Überschreitung des jeweiligen Bereichs wechseln die Anzeigen links bzw. rechts der Bereichsanzeige ihre Farbe auf gelb.



Der Fuzzy-Debugger

Über die Menüoption AUSGANGSGRÖßE kann die darzustellende Ausgangsgröße des Fuzzy Controllers gewählt werden. Die Menüoption REGELBASIS ermöglicht zu Kontrollzwecken die Ausgabe der zugrundeliegenden Regelbasis.

Parameter-dialog:



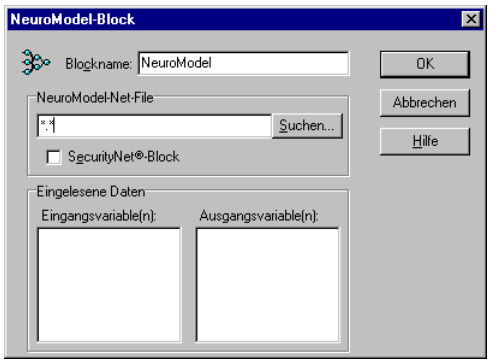
Wird für *Systemdatei* keine Extension angegeben, wird die Extension FUZ benutzt. Über die Schaltfläche *Suchen* kann ein Dateieingabedialog angefordert werden. Die zugehörigen Ein- und Ausgangsvariablen werden danach zur Kontrolle in den Listenfenstern angezeigt. Über den Schalter *Bearbeiten* kann die angegebene Datei direkt zum Bearbeiten geöffnet werden. Nach dem Verlassen des Dialogs wird die Anzahl der Blockein- und -ausgänge - sofern erforderlich - automatisch angepaßt.

NeuroModel-Block

Typname: NEUROMODEL

Funktion: Dieser Blocktyp realisiert ein mit NeuroModel® generiertes Neuronales Netz. Einzelheiten entnehmen Sie bitte der Dokumentation zu NeuroModel.

Parameter-dialog:

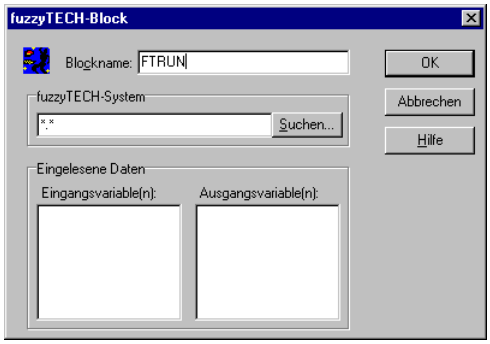


*fuzzy*TECH-Block

Typname: FTRUN

Funktion: Dieser Blocktyp realisiert einen mit *fuzzy*TECH® generierten Fuzzy Controller. Einzelheiten entnehmen Sie bitte der Dokumentation zu *fuzzy*TECH.

Parameter-dialog:

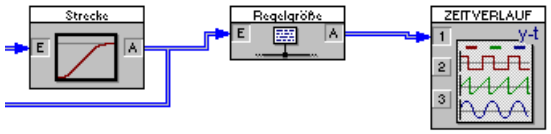


Label

Typname: LABEL

Funktion: Blöcke vom Typ *Label* haben keinerlei technische Funktion, sondern dienen lediglich dazu, Signalen einen bestimmten Namen zu geben, unter dem sie dann beispielsweise in Zeitverlaufsblöcken oder - besonders wichtig - in Superblökken auftreten (siehe Abschnitt *Arbeiten mit Superblöcken*).

Beispiel: Die Ausgangsgröße eines PT_1-Gliedes mit Namen *Strecke* soll auf einen Zeitverlaufsblock gegeben, dort aber als *Regelgröße* bezeichnet werden. Um dies zu erreichen, setzt man zwischen PT_1-Glied und Zeitverlauf ein Label mit Namen *Regelgröße* (siehe nachfolgende Grafik).



Einsatz von Label-Blöcken

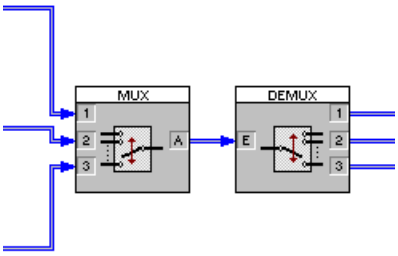
Parameterdialog:

Multiplexer

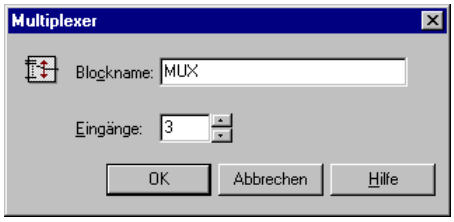
Typname: MUX

Funktion: Dieser Block ermöglicht die Zusammenfassung mehrerer Verbindungen zu einer einzigen. Sein Einsatz ist daher dann sinnvoll, wenn mehrere Verbindungen gleichzeitig über einen größeren Bereich gezogen werden können. Die Ausgangsverbindung eines Multiplexers muß mit dem Eingang eines *Demultiplexer*-Blocks verbunden werden.

Beispiel:



Parameter-dialog:

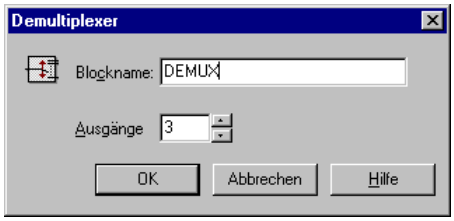


Demultiplexer

Typname: DEMUX

Funktion: Dieser Block splittet die von einem *Multiplexer*-Block erzeugte Mehrfachverbindung wieder auf. Die Eingangsverbindung eines Demultiplexers muß daher vom Ausgang eines Multiplexers stammen.

Parameter-dialog:

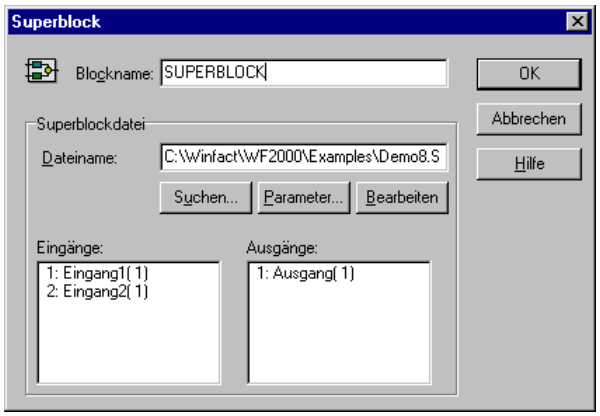


Superblock

Typname: SUPERBLOCK

Funktion: Ein Superblock enthält ein beliebiges Teilsystem aus Blöcken und Verbindungen. Der Arbeit mit Superblöcken ist ein eigener Abschnitt *Arbeiten mit Superblöcken* gewidmet.

Parameterdialog:



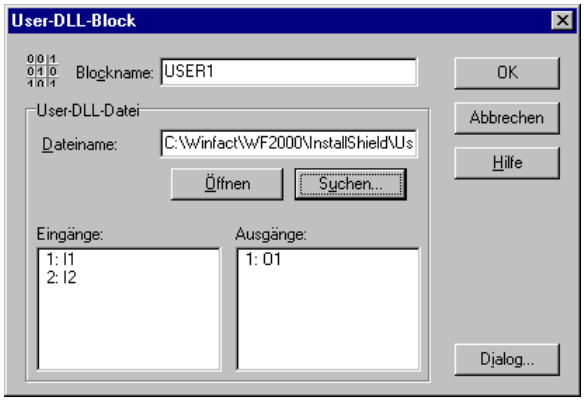
Wird für *Dateiname* keine Extension angegeben, wird die Extension SBL benutzt. Über die Schaltfläche *Suchen* kann ein Dateieingabedialog angefordert werden. Die zugehörigen Ein- und Ausgangsvariablen werden danach zur Kontrolle in den Listenfenstern angezeigt. Die Schaltfläche *Parameter...* ermöglicht die Bearbeitung der Superblock-Exportparameter. Über den Schalter *Bearbeiten* kann die angegebene Datei direkt zum Bearbeiten geöffnet werden. Nach dem Verlassen des Dialogs wird die Anzahl der Blockein- und -ausgänge - sofern erforderlich - automatisch angepaßt.

Benutzerdefinierter Block (User-DLL)

Typname: USER1

Funktion: Benutzerdefinierter Block im Windows-DLL-Format. Einzelheiten dazu finden Sie im Abschnitt *Benutzerdefinierte Systemblöcke*.

Parameter-dialog:



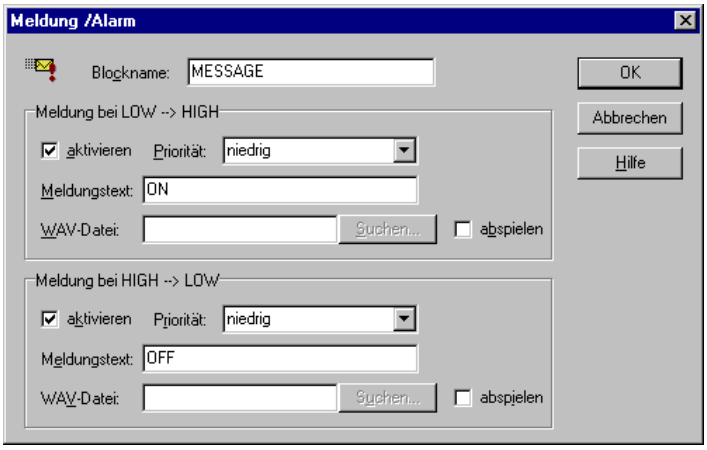
Wird für *Systemdatei* keine Extension angegeben, wird die Extension DLL benutzt. Über die Schaltfläche *Suchen* kann ein Dateieingabedialog angefordert werden. Die Schaltfläche *Öffnen* liest die angegebene Datei ein. Die zugehörigen Ein- und Ausgangsvariablen werden danach zur Kontrolle in den Listenfenstern angezeigt. Nach dem Verlassen des Dialogs wird die Anzahl der Blockein- und -ausgänge - sofern erforderlich - automatisch angepaßt. Über die Schaltfläche *Dialog...* wird der blockspezifische Parameterdialog aufgerufen.

Meldung/Alarm

Typname: MESSAGE

Funktion: Dieser Block ermöglicht die Erzeugung einer Meldung bzw. eines Alarms, sofern an seinem Eingang eine positive oder negative Flanke auftritt. Jede Meldung kann neben der grafischen Anzeige durch das Abspielen einer WAV-Datei gekennzeichnet werden. Einzelheiten dazu siehe im Abschnitt *Verwaltung von Meldungen und Alarmen*.

Parameter-dialog:



Hardware-Interface

Typname: HARDWARE

Funktion: Dieser Block dient als Interface zu Hardware jeglicher Art (z. B. PC-Einsteckkarten, externe Hardware am RS-232-Port etc.) und kann sowohl Hardware-Eingänge als auch Hardware-Ausgänge enthalten. Hinweise dazu entnehmen Sie bitte der Dokumentation zum jeweiligen Hardware-Treiber.

Arbeiten mit Superblöcken

Was ist ein Superblock?

Ein Superblock ist ein spezieller Systemblocktyp, der durch *Gruppierung mehrerer Systemblöcke und ihrer Verbindungen* entsteht. Der Superblock stellt also nichts anderes als ein Teilsystem dar, das zu einem neuen Block - in der Regel mit Ein- und Ausgängen - zusammengefaßt wird. Von außen betrachtet hat der Superblock dann eine Art "Black-Box"-Charakteristik. Superblöcke eignen sich damit insbesondere für die übersichtliche Strukturierung komplexer Systeme sowie zur Gruppierung häufig benutzter Teilsysteme.

Beispiel: Sie möchten für einen zuvor modellierten Prozeß - der beispielsweise aus der Reihenschaltung dreier Systemblöcke besteht - verschiedene Regler testen. Dann wandeln Sie zunächst die Reihenschaltung in einen Superblock um, auf den Sie dann später aus beliebigen Systemen zugreifen können.

Wie werden Informationen über den internen Aufbau des Superblocks abgelegt?

Superblöcke in BORIS sind *dateireferenziert*: Alle Informationen über die im Superblock enthaltenen Blöcke und Verbindungen werden in einer Datei mit der Extension SBL abgelegt. Diese Dateien sind - bis auf einen in Superblockdateien zusätzlich enthaltenen Dateiheader - mit "normalen" BORIS-Systemdateien vom Typ BSY identisch. Somit lassen sich Superblockdateien nach dem Laden auch als gewöhnliche Systemdateien speichern und umgekehrt, ebenso können sie natürlich eigenständig simuliert werden.

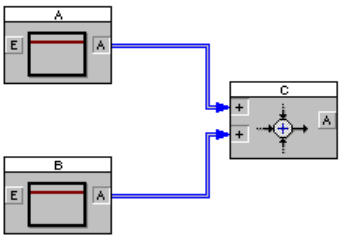
Der Vorteil der Dateireferenzierung liegt auf der Hand: Wird in einer Superblockdatei eine Änderung vorgenommen, so sind automatisch alle Systemdateien, die diesen Superblock einbinden, aktualisiert. Der Superblock selbst ist über die Angabe seines Dateinamens eindeutig definiert.

Ein- und Ausgänge von Superblöcken

Die Ein- und Ausgänge eines Superblocks werden von BORIS automatisch festgelegt. Dabei gilt es folgendes zu beachten:

- Alle offenen Systemblockein- bzw. -ausgänge werden zu Ein- bzw. Ausgängen des Superblocks.

  Beispiel: Nachfolgende Struktur soll zu einem Superblock gruppiert werden.

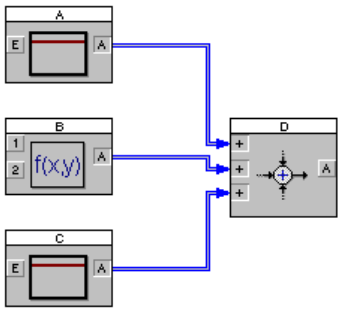
  

  Der resultierende Superblock besitzt zwei Eingänge (die der Blöcke A und B) sowie einen Ausgang (den von Block C).

- Sollen Blockausgänge, die bereits eine Verbindung enthalten (zum Beispiel eine Rückführung), zu Superblockausgängen werden, so setzt man zweckmäßigerweise Label-Blöcke ein (siehe Abschnitt *Superblöcke und Labels*). Sollen im umgekehrten Fall offene Systemeingänge nicht zu Superblockeingängen werden, so beschalten Sie diese einfach z. B. mit einem Konstanten-Block, der auf null (oder einen anderen geeigneten Wert) gesetzt wird.

- Die Ein- bzw. Ausgänge des Superblocks werden in der Reihenfolge durchnumeriert, in der die entsprechenden Blöcke eingefügt wurden.

  Beispiel:

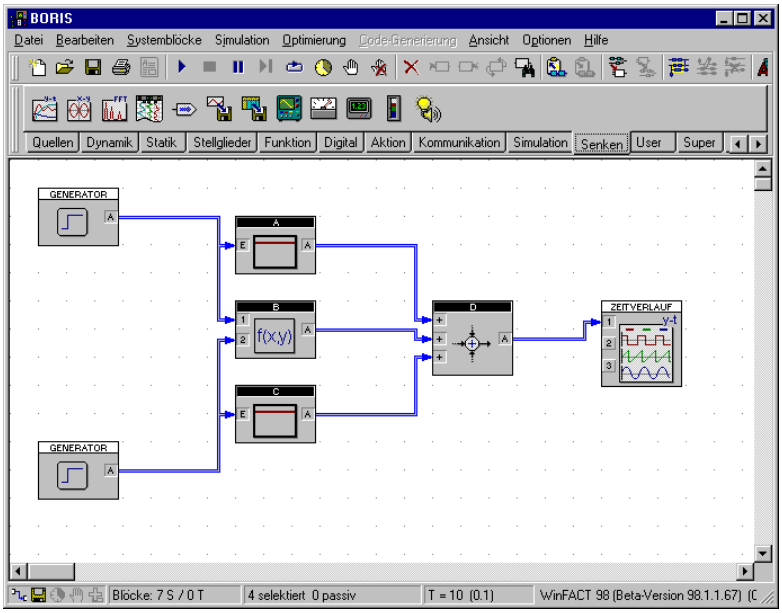
Wurden die Blöcke in der Reihenfolge A, B, C, D eingefügt, so wird der Eingang von Block A zum ersten, der erste Eingang von Block B zum zweiten, der zweite Eingang von Block B zum dritten und der Eingang von Block C zum vierten Superblockeingang. Analoges gilt bei Vorliegen mehrerer Ausgänge. Sollen Superblockein- und -ausgänge vertauscht werden, kann man entweder die entsprechenden Systemblöcke zunächst löschen und dann wieder in geeigneter Reihenfolge einfügen oder aber *Labels* einfügen (siehe Abschnitt *Superblöcke und Labels*).

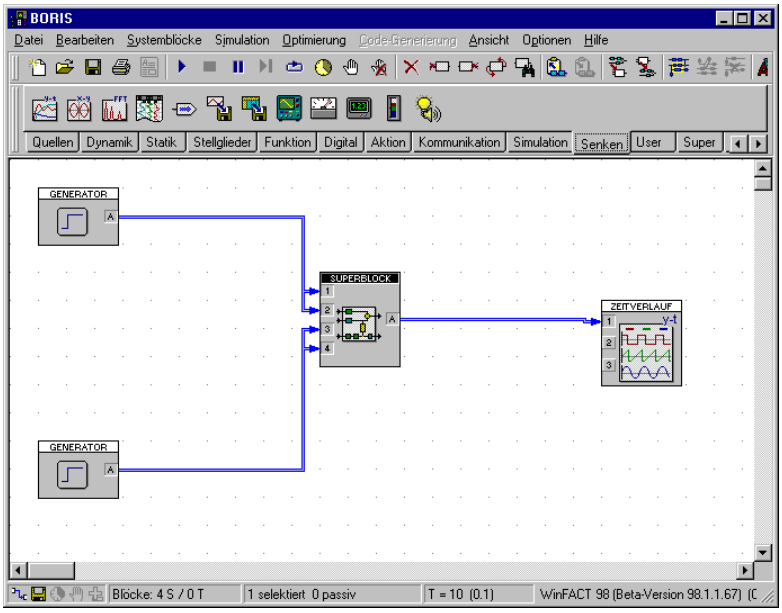
Um einen neuen Superblock zu definieren...

... haben Sie grundsätzlich zwei Möglichkeiten:

- Sie erstellen den Superblock zunächst als separate Datei, speichern ihn über DATEI | DATEI SPEICHERN UNTER... ab und laden ihn dann später über die Angabe des Dateinamens in das entsprechende übergeordnete System.
- Soll ein Teilsystem einer bereits konfigurierten Systemstruktur in einen Superblock überführt werden, so selektieren Sie zunächst die zu gruppierenden Blöcke und nehmen dann die eigentliche Gruppierung über BEARBEITEN | GRUPPIEREN ZU SUPERBLOCK oder die Schaltfläche  der System-Toolbar vor. BORIS fragt Sie dann nach dem Namen, den die Superblockdatei haben soll und fügt anschließend den Superblock ein (siehe nachfolgende Grafiken).

Ein Superblock läßt sich durch Selektieren und BEARBEITEN | SUPERBLOCK AUFLÖSEN oder die Schaltfläche  jederzeit wieder in seine Bestandteile zerlegen. Dabei ist darauf zu achten, daß im "Umkreis" des Superblocks genügend Platz zur Verfügung steht.



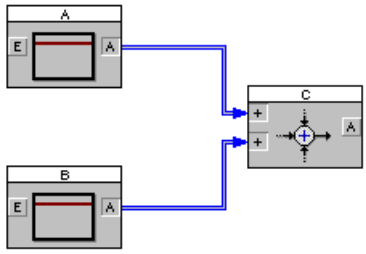


Gruppieren von Blöcken zu einem Superblock: Selektieren der Blöcke (oben) und Bildschirm nach Umwandlung in einen Superblock (unten)

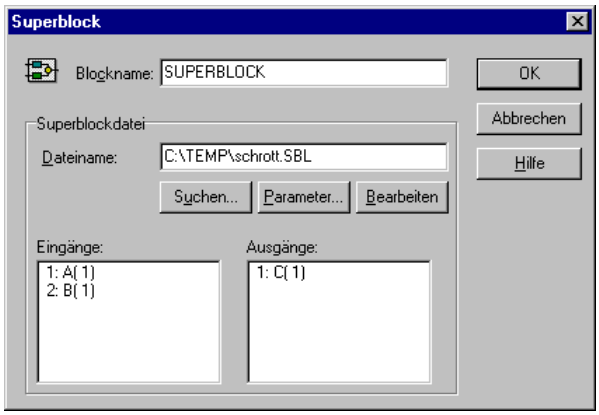
Superblöcke und Labels

Der Einsatz von Label-Blöcken kann im Zusammenhang mit Superblöcken an verschiedenen Stellen sinnvoll sein:

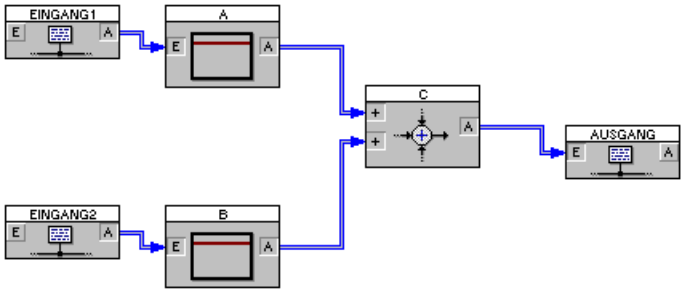
- Zur Umbenennung der Superblockein- bzw. -ausgänge. Damit der Anwender weiß, welcher Ein- bzw. Ausgang des Superblocks welche Funktion hat, auch ohne daß er die Superblockdatei zunächst zur Betrachtung laden muß, erhalten alle Superblockein- und -ausgänge einen Namen, der im Parameterdialog des Superblocks angezeigt wird. Als Voreinstellung wird der Name des intern mit dem Ein- bzw. Ausgang verbundenen Blocks in Kombination mit dessen Ein- bzw. Ausgangsnummer benutzt. Betrachten Sie dazu folgendes Beispiel:



Wird dieses Teilsystem in einen Superblock verwandelt, so werden die Eingänge des Superblocks mit *A(1)* bzw. *B(1)* bezeichnet, der Ausgang mit *C(1)*. Der Parameterdialog des Superblocks sieht dann also wie folgt aus:



Um den Ein- und Ausgängen griffigere Namen zu geben, kann man ihnen nun Labels vor- bzw. nachschalten, die dann mit den gewünschten Namen bezeichnet werden. Die modifizierte Superblockstruktur sieht dann wie folgt aus:



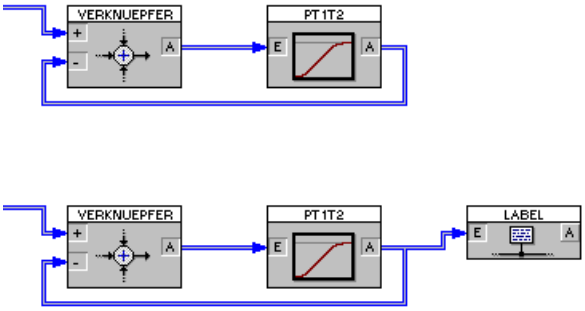
Die Eingänge wurden hier mit *Eingang1* bzw. *Eingang2* bezeichnet, der Ausgang mit *Ausgang* (in der Praxis sollte man möglichst sinnvollere Namen wählen!). Ruft man nunmehr den Parameterdialog des Superblocks auf, erscheinen dort die entsprechenden Einträge.

Ändern der Reihenfolge

- Für die Umnumerierung von Ein- oder Ausgängen. Sollen die Superblockein- oder -ausgänge eine neue Reihenfolge erhalten, fügt man einfach entsprechende Labels in der gewünschten Reihenfolge ein.

Erzeugung offener Ausgänge

- Zur Erzeugung offener Ausgänge. Soll ein Ausgang der Superblockstruktur, der bereits eine Verbindung enthält, zu einem Superblockausgang werden, so setzen Sie hinter den Ausgang ein Label und erzeugen so einen offenen Ausgang. Nachfolgende Grafik zeigt ein derartiges Beispiel.



Um den Ausgang des PT1T2-Glieds (oben) zu einem Superblockausgang zu machen, wird ihm ein Label nachgeschaltet (unten).

Ausgangsblöcke in Superblöcken

Alle Ausgangsblöcke (z. B. Zeitverlauf, Oszillograph usw.) können ohne weiteres auch innerhalb von Superblöcken verwendet werden. Dabei ist jedoch zu beachten, daß das entsprechende Anzeigefenster *nur während der Simulation* sichtbar ist! Es erscheint also bei Start der Simulation und verschwindet unmittelbar nach Beendigung der Simulation. Eine "Nachbetrachtung" der Simulationsergebnisse ist in diesen Fällen also nicht möglich. Das gleiche gilt auch für eventuelle Anzeigefenster anderer Blöcke (z. B. den Fuzzy-Debugger des Fuzzy Controllers).

Quellen und Senken in Superblöcken

Auch Signalquellen und -senken können in Superblöcken wie gewohnt verwendet werden. In der Regel wird man beide Blocktypen dabei in der Betriebsart *lokal* betreiben, so daß Ihre Gültigkeit auf den jeweiligen Superblock beschränkt bleibt. Prinzipiell können aber auch *globale* Quellen und Senken in einem Superblock benutzt werden. Dies bedeutet einerseits, daß ein Signal, das von einem Superblock über eine *globale* Signalsenke versendet wird, auch außerhalb des Superblocks mit Hilfe einer zugehörigen *globalen* Signalquelle empfangen werden kann. Andererseits können Superblöcke über interne, *globale* Signalquellen alle Signale empfangen, die von *globalen* Signalsenken außerhalb des Blocks generiert werden.

Vorsicht: Wird derselbe Superblock mehrmals in einer Struktur eingesetzt, sollte auf *globale* Signalsenken innerhalb des Superblocks in jedem Fall verzichtet werden, da ansonsten mehrere Senken mit demselben Namen existieren! Dies führt bei der Simulation in der Regel zu unerwünschten "Effekten"!

Superblöcke in Superblöcken in Superblöcken...

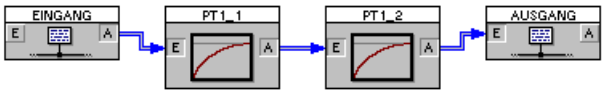
Ein System kann beliebig viele Superblöcke enthalten. Ein Superblock selbst kann ebenfalls einen oder mehrere Superblöcke enthalten usw. - die Verschachtelungstiefe ist beliebig. Nicht erlaubt sind selbstverständlich rekursive

Referenzen von Superblöcken: Enthält Superblock A den Superblock B, so darf letzterer nicht gleichzeitig wieder auf Block A zugreifen.

Exportieren von Parametern

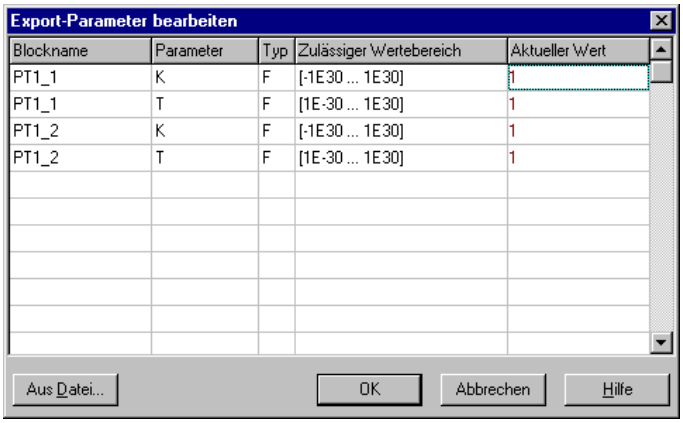
Häufig möchte man zwar dieselbe Struktur mehrfach verwenden, jedoch jedes Mal mit unterschiedlichen Parametern. BORIS bietet für solche Zwecke die Möglichkeit, die Exportparameter von Blöcken "nach außen" zu führen, so daß sie von der nächsthöheren Systemebene modifiziert werden können. Dazu sind lediglich innerhalb der betreffenden Superblockdatei die entsprechenden Exportparameter zu aktivieren; sie können dann - unabhängig von ihrem Wert innerhalb der Superblockdatei – bei jeder Nutzung des Superblocks unabhängig gesetzt werden.

Beispiel: Ein einfacher Superblock bestehe aus der Reihenschaltung zweier PT1-Glieder (siehe nachfolgende Grafik). Dieser Superblock soll nun von anderen Strukturen benutzt werden, wobei Verstärkungen und Zeitkonstanten jeweils innerhalb der aufrufenden Struktur modifiziert werden sollen. Dazu müssen lediglich in den Parameterdialogen der beiden PT1-Glieder jeweils die Exportparameter aktiviert werden (siehe auch Beispieldatei SUPEREXP.BSY im Examples-Verzeichnis).



Superblock mit Exportparametern

Nach dem Speichern läßt sich der Superblock wie gewohnt in eine beliebige Systemstruktur einbinden, wobei nunmehr aber jeweils die exportierten Parameter *für jeden Superblock getrennt* modifiziert werden können. Dazu dient die Schaltfläche *Parameter...* innerhalb des Superblock-Parameterdialogs. Die hier vorgegebenen Parameter werden dann nicht innerhalb der Superblockdatei, sondern innerhalb der aufrufenden Datei gespeichert. Da die Zuordnung der Exportparameter anhand des Blocknamens vorgenommen wird, ist es wichtig, alle exportierenden Blöcke mit eindeutigen Namen zu versehen. Dies kann vorab über DATEI | AUF DOPPELTE BLOCKNAMEN ÜBERPRÜFEN... überprüft werden.



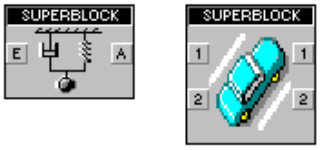
Zugehöriger Dialog zur Modifikation der Superblock-Exportparameter

Beachten Sie bitte, daß der Export von Superblock-Parametern *nur in die nächsthöhere* Ebene, nicht aber darüber hinaus möglich ist. Wird z. B. ein Superblock A von einem Superblock B benutzt, so sind dort die Exportparameter des Superblocks A modifizierbar. Wird jedoch Superblock B seinerseits in eine Struktur eingebunden, so kann von dort aus auf die Exportparameter von Superblock A nicht mehr zugegriffen werden!

Benutzerdefinierte Block-Bitmaps

Optional hat der Anwender die Möglichkeit, jeden seiner Superblöcke mit einem *blockspezifischen Bitmap* zu versehen. Dieses Block-Bitmap wird als BMP-Datei erstellt und muß denselben Namen aufweisen wie die Superblockdatei selbst, jedoch mit der Extension BMP, und sich im selben Verzeichnis befinden wie der Superblock. Zu einem Superblock mit dem Namen MOTOR.SBL gehört also die Bitmap-Datei MOTOR.BMP. BORIS überprüft beim Einfügen eines Superblocks automatisch, ob die zugehörige BMP-Datei vorhanden ist. Ist dies der Fall, wird diese benutzt, ansonsten das Standard-Superblock-Bitmap. Das Bitmap sollte eine Größe von 48×42 Pixeln aufweisen; Bitmaps anderer Größe werden automatisch gedehnt bzw. gestaucht, so daß sie optimal in das Blockschaltbild passen. Auch für die Druckerausgabe kann ein benutzerdefiniertes Bitmap (möglichst als s/w-Bitmap) definiert werden. Dieses muß am Ende des Dateinamens zusätzlich die Kennung *_P* erhalten.

Beispiele: zu MOTOR.SBL gehört Drucker-Bitmap MOTOR_P.BMP
zu GETRIEBE.SBL gehört Drucker-Bitmap GETRIEBE_P.BMP



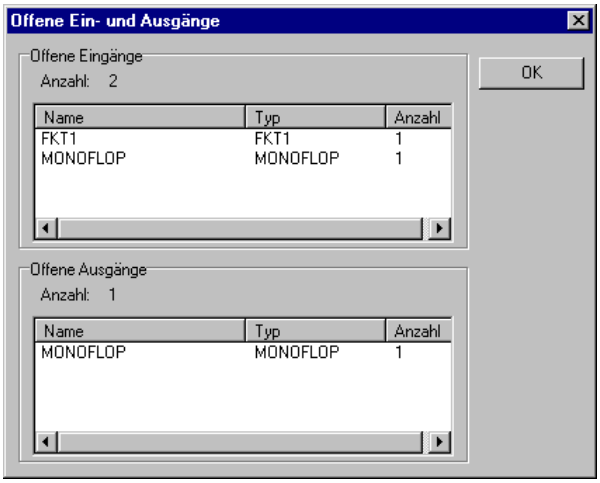
Beispiel für Superblöcke mit anwenderspezifischem Block-Bitmap

Schließlich kann auch für die Darstellung des Superblocks in der Palette *Super* der Systemblock-Toolbar ein Bitmap definiert werden. Dieses hat die Größe von 18x18 Pixeln und den gleichen Namen wie das Drucker-Bitmap, jedoch die Kennung *_T*.

Beispiele: zu MOTOR.SBL gehört Toolbar-Bitmap MOTOR_T.BMP
zu GETRIEBE.SBL gehört Toolbar-Bitmap GETRIEBE_T.BMP

Was sonst noch wissenswert ist

Um sich einen Überblick zu verschaffen, wieviele offene Ein- und Ausgänge das aktuelle System besitzt, dient die Menüfolge DATEI | OFFENE EIN-/AUSGÄNGE... Es erscheint ein Dialog, der jeweils Blocknamen, Blocktyp und die Anzahl der offenen Ein- bzw. Ausgänge auflistet.



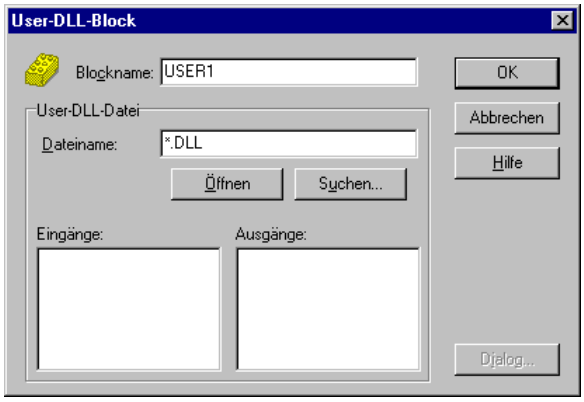
Dialog zur Auflistung offener Ein- und Ausgänge

In einigen Fällen kann es passieren, daß einige Eingänge eines Teilsystems, das zu einem Superblock gruppiert werden soll, offen sind, aber nicht benötigt werden (z. B. Steuereingänge o. ä.) und daher auch nicht als Eingänge des Superblocks auftauchen sollen. In diesen Fällen sollte man diese Eingänge einfach mit einem Konstanten-Block beschalten, der einen geeigneten Wert (in der Regel wird dies 0 sein) aufschaltet.

Benutzerdefinierte Systemblöcke

Das Konzept der User-DLLs

BORIS erlaubt die Programmierung eigener Systemblocktypen, sog. *User-DLL-Blöcke*, auf Basis einer 32-Bit-Windows-DLL. Diese lassen sich mit praktisch jeder 32-Bit-Programmierumgebung wie z. B. Delphi 3/4, Visual Basic 5 oder Visual C++ erstellen. Für den programmierunerfahrenen Anwender steht dazu der optional erhältliche *BORIS-User-DLL-Experte* zur Verfügung.



Parameterdialog eines User-DLL-Blocks (hier noch leer!)

Im folgenden werden zunächst die *Datenschnittstelle* und die *Funktionsschnittstelle* des User-Blocks besprochen, bevor dann anhand von Beispielen unterschiedlicher Komplexität eine tiefergehende Einführung in die Programmierung von User-Blöcken erfolgt.

Hinweis: Alle nachfolgend abgedruckten Programme bzw. Programmsegmente wurden mit Delphi 3 erstellt. Eine Übertragung auf andere Entwicklungsumgebungen oder Programmiersprachen ist aber ohne größere Probleme möglich.

Die Datenschnittstelle des User-Blocks

Alle für den Anwender relevanten Daten eines User-Blocks sind in einer Datenstruktur vom Typ *TParameterStruct* bzw. einem Zeiger *PParameterStruct* auf diese Struktur festgelegt. Diese Datenstruktur enthält zunächst die eigentlichen *Blockparameter*. Diese Daten sind in der Regel blockspezifische Konstanten (z. B. Verstärkungsfaktoren, Zeitkonstanten oder ähnliches) und können vom Anwender über den Parameterdialog des User-Blocks geändert werden. Sie werden beim Speichern einer BORIS-Struktur mit dem User-Block abgespeichert und stehen damit nach dem erneuten Einlesen der Datei wieder zur Verfügung. Darüber hinaus können diese Parameter natürlich auch für andere Zwecke - z. B. als Zustandsvariablen, Zwischen- oder Hilfsvariablen irgendwelcher Art, Flags usw. - "mißbraucht" werden.

Folgende Parameter stehen zur Verfügung:

- Bis zu 32 Fließkommazahlen (10 Byte, Datentyp *extended* in PASCAL bzw. *long double* in C)
- Bis zu 32 Integer-Zahlen (4 Byte, Datentyp *longint* bzw. *integer* in Pascal bzw. *int* in C)
- Bis zu 32 Schalter- bzw. Byte-Variablen (1 Byte, Datentyp *byte* in Pascal bzw. *char* in C)
- Eine Stringvariable der Länge 256 (z. B. für Dateinamen)

Für die Fließkomma- bzw. Integer-Parameter können Minimal- und Maximalwerte vorgegeben werden, deren Einhaltung dann im Parameterdialog automatisch überprüft wird. Für alle Parameter können bzw. müssen außerdem Namen vergeben werden, die dann an entsprechender Stelle im Parameterdialog erscheinen. Weitere Parameter können bei Bedarf auch aus einer zusätzlichen

Datei gelesen werden. Neben diesen eigentlichen Blockparametern enthält die Datenstruktur *TParameterStruct* einige weitere Variablen, die nur für fortgeschrittene Anwendungen benötigt werden und an späterer Stelle im Rahmen der Beispiele genauer erläutert werden. Zunächst folgt eine knappe Auflistung der Komponenten von *TParameterStruct*.

```
type
  PParameterStruct = ^TParameterStruct;
  TParameterStruct = packed record
    NuE: Byte;                     {Anzahl Fließkomma-Parameter}
    NuI: Byte;                     {Anzahl Integer-Parameter}
    NuB: Byte;                     {Anzahl Schalter-Parameter}
    E: Array[0..31] of Extended;   {Fließkomma-Parameter}
    I: Array[0..31] of LongInt;    {Integer-Parameter}
    B: Array[0..31] of Byte;       {Schalter-Parameter}
    D: Array[0..255] of char;      {Dateiname für optionale Daten}
    EMin: Array[0..31] of Extended; {unt. Grenze Fließkomma-Par.}
    EMax: Array[0..31] of Extended; {obere Grenze Fließkomma-Par.}
    IMin: Array[0..31] of LongInt;  {untere Grenze Integer-Par.}
    IMax: Array[0..31] of LongInt;  {obere Grenze Integer-Par.}
    NaE: Array[0..31,0..40] of char; {Namen Fließkomma-Parameter}
    NaI: Array[0..31,0..40] of char; {Namen Integer-Parameter}
    NaB: Array[0..31,0..40] of char; {Namen Schalter-Parameter}
    UserDataPtr: Pointer;            {Zeiger auf opt. Variablen}
    ParentPtr: Pointer;              {Zeiger auf User-DLL-Block}
    ParentHWnd: Word;                {BORIS-Fensterhandle}
    ParentName: PChar;               {Name des User-DLL-Blocks}
    UserHWindow: Word;               {Benutzerdef. Fensterhandle,
                                      z. B. für Ausgabefenster}
    DataFile: text;                  {Optionale Textdatei}
  end;
```

Struktur der Datenschnittstelle TParameterStruct *in Pascal*

Variable	Bedeutung
NuE, NuI, NuB	Enthält die Anzahl der Fließkomma-, Integer- bzw. Schalterparameter des Blocks. Dabei sind nur die wirklich als Blockparameter benutzten Komponenten zu berücksichtigen, die im Parameterdialog erscheinen sollen, nicht aber die als Hilfsgrößen (z. B. Zustandsvariablen, Flags o. ä.) verwendeten Komponenten der Arrays *E*, *I* bzw. *B*!
E, I, B	Diese Arrays enthalten die Parameter selbst.
D	Enthält bei Bedarf den Namen einer externen Datei für das Einlesen weiterer Daten.
EMin, EMax	Diese Arrays enthalten die unteren bzw. oberen zulässigen Werte für die Fließkomma-Parameter

IMin, IMax	Diese Arrays enthalten die unteren bzw. oberen zulässigen Werte für die Integer-Parameter.
NaE, NaI, NaB	Arrays mit den Namen der Fließkomma-, Integer- bzw. Schaltervariablen. Jeder Name darf maximal 40 Zeichen aufweisen
UserDataPtr	Zeiger auf optionale Blockvariablen (z. B. Zustandsgrößen, Zwischenwerte oder ähnliches). Hierunter sind in der Regel solche Variablen zu verstehen, die keine Blockparameter sind und damit auch nicht mit dem Block abgespeichert werden müssen.
ParentPtr	Dieser Zeiger zeigt auf den User-DLL-Block. Wird für Anwender-DLLs im allgemeinen nicht benötigt.
ParentHWnd	Handle des BORIS-Hauptfensters. Wichtig bei User-DLLs, die eigene Fenster (z. B. zur Visualisierung) oder Dialoge besitzen.
ParentName	Zeiger auf den Namen des User-DLL-Blocks. Wird i. a. nur benötigt bei User-DLL-Blöcken mit Visualisierungsfunktion
UserHWindow	In dieser Variablen kann z. B. das Fensterhandle eines mit dem User-Block generierten Visualisierungsfensters gespeichert werden. Wird i. a. nur benötigt bei User-DLL-Blöcken mit Visualisierungsfunktion.
DataFile	Textdatei für universelle Zwecke. Kann zusammen mit der Variablen *D* (s. o.) verwendet werden.

Neben der wichtigsten Datenstruktur *TParameterStruct* müssen in der User-DLL einige weitere Datenstrukturen deklariert sein. Dazu gehört zunächst die Struktur *TDialogEnableStruct* bzw. der zugehörige Zeiger *PDialogEnableStruct*.

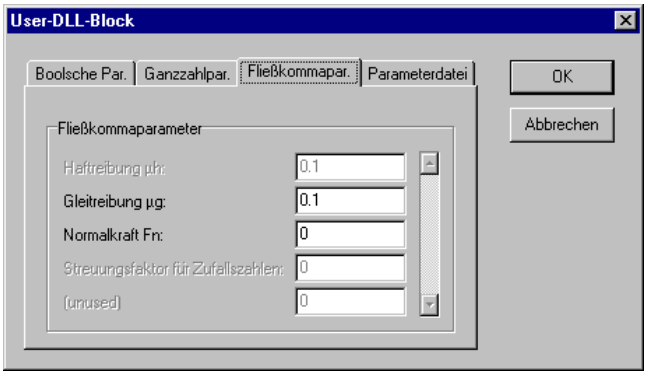
```
type
  PDialogEnableStruct=^TDialogEnableStruct;
  TDialogEnableStruct=record
    AllowE: Longint;          { Soll die Eingabe eines Wertes   }
    AllowI: Longint;          { un-/zulässig sein so ist das Bit}
    AllowB: Longint;          { des Allow?-Feldes 0 bzw. 1}
    AllowD: Byte;
  end;
```

Struktur von TDialogEnableStruct *(in Pascal)*

Diese Struktur wird zur Verwaltung des Parameterdialogs benötigt; sie ermöglicht im Zusammenspiel mit der Prozedur *GetDialogEnableStruct* (siehe später) ein bedingtes Aktiv- bzw. Passivsetzen von Dialogelementen (Beispiel: Ein

Fließkomma-Parameter soll nur modifizierbar sein, wenn ein bestimmter Schalter im Dialog gesetzt ist). Dazu enthält *TDialogEnableStruct* für jedes der jeweils 32 Eingabefelder eines Variablentyps ein entsprechendes Bit, das gesetzt (Eingabefeld wird aktiv) bzw. gelöscht (Eingabefeld wird passiv) wird. Das Feld *AllowD* legt schließlich fest, ob der Name der optionalen Parameterdatei (Variable *D* aus *TParameterStruct*) aktiv ist.

Nachfolgende Grafik zeigt beispielhaft die Palettenseite *Fließkommaparameter* des Parameterdialogs eines User-DLL-Blocks zur Reibungssimulation, der sich unter dem Namen REIBUNG.DLL im Verzeichnis *UserDLLs* (Quelltext REIBUNG.DPR im Unterverzeichnis *Sources*) befindet. Hier wurden u. a. vier Fließkomma-Parameter definiert, nur der zweite und dritte sind z. Z. aktiv.



Beispiel für den Parameterdialog eines User-Blocks mit vier Fließkomma-Parametern

Die letzten erforderlichen Datenstrukturen betreffen die Definition der Ein- und/oder Ausgänge des User-Blocks. Zunächst ist die Struktur *TNumberOfInputsOutputs* zu deklarieren. In dieser Struktur sind die Anzahl der Ein- bzw. Ausgänge des Blocks sowie ihre Namen (jeweils max. 40 Zeichen) festgelegt:

```
type
  PNumberOfInputsOutputs = ^TNumberOfInputsOutputs;
  TNumberOfInputsOutputs = packed record
    Inputs  :Byte;                        {Anzahl Eingänge}
    Outputs :Byte;                        {Anzahl Ausgänge}
    NameI : Array[0..49] of String[40];   {Namen der Eingänge}
    NameO : Array[0..49] of String[40];   {Namen der Ausgänge}
  end;
```

Struktur von TNumberOfInputsOutputs *(in Pascal)*

Schließlich bleibt die Deklaration der Datentypen für die spätere Übergabe der aktuellen Blockein- und -ausgangswerte. Dazu dienen die Datentypen *TInputArray* bzw. *TOutputArray*:

```
type
  PInputArray = ^TInputArray;
  TInputArray = packed Array[1..50] of extended;   {Eingangswerte}
  POutputArray = ^TOutputArray;
  TOutputArray = packed Array[1..50] of extended;  {Ausgangswerte}
```

Datentypen TInputArray *bzw.* TOutputArray *(in Pascal)*

Damit sind alle erforderlichen Typendeklarationen erläutert. Die Deklaration in C erfolgt analog; nachfolgendes Listing faßt alle Datentypen dafür zusammen.

```
typedef struct{
    char NuE;
    char NuI;
    char NuB;
    long double E[32];
    int  I[32];
    char B[32];
    char D[256];
    long double EMin[32];
    long double EMax[32];
    int  IMin[32];
    int  IMax[32];
    char NaE[32][41];
    char NaI[32][41];
    char NaB[32][41];
    void FAR *UserDataPtr;
    void FAR *ParentPtr;
    unsigned int ParentHWnd;
    char *ParentName;
    unsigned int UserHWindow;
    FILE *DataFile
  } TParameterStruct, FAR *PParameterStruct;

typedef struct {
    long AllowE;
    long AllowI;
    long AllowB;
    char AllowD;
  } TDialogEnableStruct, FAR *PDialogEnableStruct;

typedef struct {
    char Inputs;
    char Outputs;
    char NameI[50][41];
    char NameO[50][41];
  } TNumberOfInputsOutputs, FAR *PNumberOfInputsOutputs;

typedef long double TInputArray[50];
typedef long double TOutputArray[50];
```

Datenschnittstelle des User-Blocks in der Sprache C

Die Funktionsschnittstelle des User-Blocks

Die Funktionsschnittstelle der User-DLL enthält die für den User-Block zu definierenden Funktionen bzw. Prozeduren. Diese unterscheiden sich in solche Funktionen, die zur Verwaltung des User-Blocks bzw. seines Parameterdialogs erforderlich sind (*organisierende* Funktionen) und die Funktionen, die die eigentliche Funktionalität des Blocks beschreiben, d. h. während der Simulation bzw. unmittelbar davor oder danach aufgerufen werden (*simulierende* Funktionen). Folgende Auflistung gibt zunächst eine Übersicht über alle erlaubten Funktionen mit ihren Parametern, bevor dann die Bedeutung aller Funktionen im Detail erläutert wird. Alle Funktionen sind als *far pascal* zu deklarieren.

Organisierende Funktionen:

```
Procedure IsUserDLL32;                                          (*)

Procedure GetParameterStruct(D:PParameterStruct);               (*)

Procedure GetDialogEnableStruct(D:PDialogEnableStruct;
                                D2:PParameterStruct);           (*)

Procedure GetNumberOfInputsOutputs(D:PNumberOfInputsOutputs);   (*)
Procedure GetNumberOfInputsOutputs2(D:PNumberOfInputsOutputs;
                                    ParameterFileName: PChar;
                                    UserDataPtr: Pointer;
                                    var UserHWindow: THandle);

Procedure InitUserDLL(D:PParameterStruct);
Procedure DisposeUserDLL(D:PParameterStruct);

Procedure InitUserData(D: PParameterStruct);
Procedure DisposeUserData(D: PParameterStruct);

Procedure DialogOK(D:PParameterStruct);

Procedure ShowWindowDLL(D: PParameterStruct);
Procedure HideWindowDLL(D: PParameterStruct);

Function  SetInputChar: PChar;
Function  SetOutputChar: PChar;

Function  GetDLLName: PChar;

Procedure WriteToFile(AFileHandle: word; D: PParameterStruct);
Procedure ReadFromFile(AFileHandle: word; D: PParameterStruct);
Function  NumberOfLinesInSystemFile: word;

Procedure CallParameterDialogDLL(D1: PParameterStruct;
                                 D2: PNumberOfInputsOutputs);
```

Simulierende Funktionen:

```
Function  CanSimulateDLL(D:PParameterStruct):Integer;
```

```
Procedure InitSimulationDLL(D:PParameterStruct;
                            Inputs:PInputArray;
                            Outputs:POutputArray);

Procedure SimulateDLL(T:Extended; D:PParameterStruct;
                      Inputs:PInputArray;
                      Outputs:POutputArray);
Procedure SimulateDLL2(T, DeltaT:Extended; D:PParameterStruct;
                       Inputs:PInputArray;
                       Outputs:POutputArray);

Procedure EndSimulationDLL;
Procedure EndSimulationDLL2(D: PParameterStruct);

Procedure SetEnhancedInformation(DeltaT, TSimu: extended;
                                 D: PParameterStruct);
```

Hinweis: Die mit einem Stern (*) gekennzeichneten Funktionen müssen in jeder User-DLL *auf jeden Fall* exportiert werden! Sofern Sie diese Funktionen nicht benötigen, lassen Sie die entsprechenden Funktionskerne einfach leer.

Bei der nachfolgenden Beschreibung der einzelnen Funktionen ist jeweils sowohl die Pascal-Signatur (oben) als auch die C-Signatur angegeben.

IsUserDLL32

Signatur

```
procedure IsUserDLL32; export far pascal;

void far pascal IsUserDLL32
```

Funktion Diese Dummy-Funktion wird lediglich exportiert, um die DLL als BORIS-32-Bit-User-DLL zu kennzeichnen. Der eigentliche Funktionsrumpf kann leer bleiben.

GetParameterStruct

Signatur

```
procedure GetParameterStruct(D:PParameterStruct);
                                  export far pascal;

void far pascal GetParameterStruct(PParameterStruct D)
```

Parameter *D* ist ein Zeiger auf die Struktur *TParameterStruct.*

Funktion In *GetParameterStruct* legen Sie die Blockparameter sowie die Daten zum Initialisieren des Parameterdialogs fest. Es werden Namen für Parameter vergeben, die Parameter selber werden initialisiert, obere und untere Eingabegrenzen eines Parameters festgelegt und letztlich wird die Anzahl der Parameter angegeben.

Beispiel:

Nachfolgendes Beispiel definiert einen User-Block mit vier Fließkomma-Parametern mit den Namen *Haftreibung*, *Gleitreibung*, *Normalkraft*, *Streuungsfaktor* sowie zwei Schalter-Parametern mit den Namen *Reale Reibung* bzw. *Stick and Slip aus Datei*. Außerdem wird eine zusätzliche Parameterdatei mit der Extension *SIM* zugelassen.

```
procedure GetParameterStruct(D:PParameterStruct); export far pascal;
var i : Integer;
begin
  D^.NuE:=4;  {Vier Fließkomma-Parameter}
  D^.NuI:=0;  {Keine Integer-Parameter}
  D^.NuB:=2;  {Zwei Schalter-Parameter}
  StrPCopy(D^.D,'*.sim');  {Dateien mit Endung SIM zulassen}
  {Initialisierung der verwendeten Daten}
  for i:=0 to 1 do begin
     D^.E[i]:=0;    {Fließkomma-Parameter 0 und 1 zu 0 initialisieren}
     D^.EMin[i]:=0; {Untere Grenze für Fließkomma-Par. 0 und 1 auf 0}
     D^.EMax[i]:=1; {Obere Grenze für Fließkomma-Par. 0 und 1 auf 1}
     D^.B[i]:=0;    {Schalter-Parameter zu 0 initialisieren}
  end;
  D^.E[2]:=0;          {Fließkomma-Parameter 2 zu 0 initialisieren}
  D^.EMin[2]:=0;       {Untere Grenze von Fließkomma-Par. 2 auf 0}
  D^.EMax[2]:=100000;  {Obere Grenze von Fließkomma-Par. 2 auf 100000}
  D^.E[3]:=0;          {Fließkomma-Parameter 3 zu 0 initialisieren}
  D^.EMin[3]:=0;       {Untere Grenze von Fließkomma-Par. 3 auf 0}
  D^.EMax[3]:=1;       {Untere Grenze von Fließkomma-Par. 3 auf 1}
  {Namensgebung max. 40 Zeichen !}
  strPCopy(D^.NaE[0],'Haftreibung:');     {Name für Fließkomma-Par. 0}
  strPCopy(D^.NaE[1],'Gleitreibung:');    {Name für Fließkomma-Par. 1}
  strPCopy(D^.NaE[2],'Normalkraft:');     {Name für Fließkomma-Par. 2}
  strPCopy(D^.NaE[3],'Streuungsfaktor:'); {Name für Fließkomma-Par. 3}
  strPCopy(D^.NaB[0],'Reale Reibung');    {Name für Schalter-Par. 0}
  strPCopy(D^.NaB[1],'"Stick and Slip" aus Datei'); {N. f. Sch.-P. 1}
end;
```

Beispiel für die Verwendung der Funktion GetParameterStruct

GetDialogEnableStruct

Signatur

```
procedure getDialogEnableStruct(D:PDialogEnableStruct;
                                D2:PParameterStruct);
                                export far pascal;

void far pascal GetDialogEnableStruct
                        (PDialogEnableStruct D,
                         PParameterStruct D2)
```

Parameter *D* ist ein Zeiger auf die Struktur *TdialogEnableStruct.*

D2 ist ein Zeiger auf die Struktur *TParameterStruct.*

Funktion *GetDialogEnableStruct* legt die zugänglichen bzw. gesperrten (grau dargestellten) Elemente des Parametrierungsdialogs fest, d. h. welche Dialogelemente vom Anwender in welcher Situation angewählt werden können und welche nicht. Die Funktion wird

bei Aufruf des Dialogs sowie *nach jeder Eingabe* aufgerufen, die vom Anwender im Dialog vorgenommen wird. Somit kann sie ggf. unmittelbar auf diese reagieren.

Beispiel:

Bei einem User-Block mit Fließkomma- und Schalter-Parametern sollen alle Fließkomma- und Schalter-Parameter jederzeit anwählbar sein. Der Dateiname für die optionale Parameterdatei soll aber nur zugänglich sein, wenn beide Schalter-Parameter gesetzt sind.

```
procedure GetDialogEnableStruct(D:PDialogEnableStruct;
                                D2:PParameterStruct); export far pascal;
begin
  D^.AllowE:=$FFFFFFFF;  {alle Fließkomma-Parameter zugänglich}
  D^.AllowB:=$FFFFFFFF   {alle Schalter-Parameter zugänglich};
  {Dateiname nur zugänglich, wenn beide Schalter-Parameter
  gesetzt sind!}
  D^.AllowD:= Byte(D2^.B[0] and D2^.B[1]);
end;
```

Beispiel für die Verwendung der Funktion GetDialogEnableStruct

Natürlich ist es auch möglich, z. B. nur ein Dialogelement vom gesperrten in den zugänglichen Zustand (oder umgekehrt) zu überführen und die anderen unbeeinflußt zu lassen. Soll z. B. der dritte Fließkomma-Parameter (also Fließkomma-Parameter 2) aktiviert werden, alle anderen Fließkomma-Parameter aber ihren aktuellen Zustand beibehalten, so lautet die entsprechende Anweisung

```
...
AllowE := AllowE or $00000004;   {Fließkomma-Parameter 2 freigeben}
...
```

GetNumberOfInputsOutputs

Signatur

```
procedure GetNumberOfInputsOutputs
          (D:PNumberOfInputsOutputs); export far pascal;

void far pascal GetNumberOfInputsOutputs
          (PNumberOfInputsOutputs D)
```

Parameter *D* ist ein Zeiger auf die Struktur *TNumberOfInputsOutputs.*

Funktion In *GetNumberOfInputsOutputs* werden Anzahl und Namen der Ein- und Ausgänge des Blocks festgelegt. Die Namen werden später in den Listboxen *Eingangsvariable(n)* und *Ausgangsvariable(n)* des User-DLL-Dialogs angezeigt.

Beispiel:

Folgendes Listing zeigt die Realisierung der Funktion für einen User-Block mit zwei Eingängen und einem Ausgang.

```
procedure GetNumberOfInputsOutputs(D:PNumberOfInputsOutputs); export far
                                                              pascal;
begin
  D^.Inputs:=2;   { Der Block soll zwei Eingänge und}
  D^.Outputs:=1;  {            einen Ausgang haben!}
  {Namensgebung der Ein- und Ausgänge}
  StrPCopy(D^.NameI[0],'Sollwert');    {Name für ersten Eingang}
  StrPCopy(D^.NameI[1],'Istwert');     {Name für zweiten Eingang}
  StrPCopy(D^.NameO[0],'Stellgröße');  {Name für Ausgang}
end;
```

Beispiel für die Verwendung der Funktion GetNumberOfInputsOutputs

GetNumberOfInputsOutputs2

Signatur

```
Procedure GetNumberOfInputsOutputs2
            (D:PNumberOfInputsOutputs;
             ParameterFileName: PChar;
             UserDataPtr: Pointer;
             var UserHWindow: THandle); export far pascal

void far pascal GetNumberOfInputsOutputs2
                  (PNumberOfInputsOutputs D,
                   char* ParameterFileName,
                   void* UserDataPtr,
                   int* UserHWindow)
```

Parameter *D* ist ein Zeiger auf die Struktur *TNumberOfInputsOutputs.*

ParameterFileName ist ein Zeiger auf das Feld *D* von *TParameterStruct.*

UserDataPtr ist ein Zeiger auf das Feld *UserDataPtr* von *TParameterStruct.*

UserHWindow ist ein Zeiger auf das Feld *UserHWindow* von *TParameterStruct.*

Funktion Diese Funktion kann alternativ zu *GetNumberOfInputsOutputs* benutzt werden, wenn die Anzahl der Ein- und Ausgänge variabel sein soll.

Beispiel:

Folgendes Listing zeigt die Realisierung der Funktion für einen User-Block, bei dem die Anzahl der Ein- und Ausgänge aus der Datei *ParameterFileName* gelesen werden sollen.

```
Procedure GetNumberOfInputsOutputs2(D:PNumberOfInputsOutputs;
           ParameterFileName: PChar; UserDataPtr: Pointer;
```



```
            var UserHWindow: THandle); export far pascal
var ParFile: TextFile;
    nIn, nOut, i: integer
begin
  AssignFile(ParFile, ParameterFileName);  //Dateinamen zuweisen
  ResetFile(ParFile);                      //Datei öffnen
  readln(ParFile, nIn, nOut);              //Anzahl Ein-/Ausgänge einlesen
  D^.Inputs  := nIn;                       //...und dem Block zuweisen
  D^.Outputs := nOut;
  {Namensgebung der Ein- und Ausgänge}
  for i:=1 to nIn  do StrPCopy(D^.NameI[i-1], 'Eingang' + IntToStr(i));
  for i:=1 to nOut do StrPCopy(D^.NameO[i-1], 'Ausgang' + IntToStr(i));
end;
```

Beispiel für die Verwendung der Funktion GetNumberOfInputsOutputs2

InitUserDLL
DisposeUserDLL

Signatur

```
procedure InitUserDLL(D:PParameterStruct);
                                          export far pascal;
procedure DisposeUserDLL(D:PParameterStruct);
                                          export far pascal;

void far pascal InitUserDLL(PParameterStruct D)
void far pascal DisposeUserDLL(PParameterStruct D)
```

Parameter *D* ist ein Zeiger auf die Struktur *TParameterStruct.*

Funktion *InitUserDLL* wird von BORIS unmittelbar nach der Initialisierung eines User-Blocks aufgerufen. Sie können diese Funktion beispielsweise nutzen, um dynamischen Speicher für Daten anzulegen, die Sie nicht in der Parameterstruktur *TParameterStruct* des Blocks unterbringen wollen oder können. Dies werden in der Regel z. B. Zustandsgrößen, Zwischenwerte irgendwelcher Art oder umfangreichere Parameterstrukturen sein, die aus einer Parameterdatei gelesen wurden. Den Zeiger auf den hier angelegten dynamischen Speicher können Sie im Parameter *UserDataPtr* von *TParameterStruct* ablegen.

Ein anderer Anwendungsfall für diese Funktion sind User-Blöcke mit Visualisierungsfenster. Bei solchen Anwendungen kann das Visualisierungsfenster des User-Blocks in dieser Funktion generiert werden (siehe dazu das Beispiel *Füllstandsregelung* an späterer Stelle!).

Das Gegenstück zu *InitUserDLL* stellt *DisposeUserDLL* dar. Diese Funktion wird unmittelbar vor der Freigabe eines User-Blocks aufgerufen. Der in *InitUserDLL* angelegte Speicher muß daher in *DisposeUserDLL* wieder freigegeben werden.

Werden die erforderlichen Daten nicht während der gesamten "Lebensdauer" des User-Blocks, sondern nur während der eigentlichen Simulation selbst benötigt (dies wird in der Regel häufiger der Fall sein), können statt der Funktionen *InitUserDLL* und *DisposeUserDLL* auch die Funktionen *InitUserData* und *DisposeUserData* (werden nachfolgend beschrieben) benutzt werden.

Die Verwendung von *InitUserDLL* und *DisposeUserDLL* ist in der Regel nur in komplexeren User-Blöcken erforderlich.

Beispiel:

Es soll ein User-Block erstellt werden, der neben den Blockparametern eine 20x20-Matrix vom Typ *Extended* sowie einen Vektor der Dimension 50 vom Typ *Integer* benötigt. Folgendes Listing zeigt die entsprechende Verwendung der Funktionen *InitUserDLL* und *DisposeUserDLL*.

```
...
...
type
  {Deklaration des Datentyps für die User-Daten}
  PUserData = ^TUserData;
  TUserData = record
    Matrix: array[1..20, 1..20] of extended;
    Vektor: array[1..50] of integer;
  end;


procedure InitUserDLL(D: PParameterStruct); export far pascal;
begin
  ...
  ...
  {Speicher für User-Daten anfordern}
  GetMem(D^.UserDataPtr, SizeOf(TUserData));
  ...
  ...
end {of InitUserDLL};


procedure DisposeUserDLL(D: PParameterStruct); export far pascal;
begin
  ...
  ...
  {Speicher für User-Daten wieder freigeben}
  FreeMem(D^.UserDataPtr, SizeOf(TUserData));
  ...
  ...
end {of DisposeUserDLL};
...
...
```

Beispiel für die Verwendung von InitUserDLL *und* DisposeUserDLL

Soll während der Simulation dann z. B. auf ein bestimmtes Matrixelement zugegriffen werden, lautet die entsprechende Pascal-Anweisung:


```
...
...
{2. Zeile, 3. Spalte der Matrix auf 5 setzen}
PUserData(UserDataPtr)^.Matrix[2, 3] := 5;
...
...
```

Zugriff auf User-Daten

InitUserData
DisposeUserData

Signatur

```
procedure InitUserData(D:PParameterStruct);
                                              export far pascal;
procedure DisposeUserData(D:PParameterStruct);
                                              export far pascal;

void far pascal InitUserData(PParameterStruct D)
void far pascal DisposeUserData(PParameterStruct D)
```

Parameter *D* ist ein Zeiger auf die Struktur *TParameterStruct.*

Funktion *InitUserData* und *DisposeUserData* entsprechen von ihrem Anwendungsbereich her den zuvor beschriebenen Funktionen *InitUserDLL* und *DisposeUserDLL*, werden aber nicht bei der Initialisierung bzw. Freigabe des User-Blocks, sondern unmittelbar vor bzw. nach der Simulation (d. h. direkt vor *InitSimulationDLL* bzw. nach *EndSimulationDLL*) aufgerufen. Sie eignen sich damit besonders für die Verwaltung von Daten, die nur während der Simulation selbst benötigt werden (siehe dazu das Beispiel *Kennfeld-DLL* an späterer Stelle). Auch diese Funktionen sind für die Realisierung einfacher DLLs nicht erforderlich.

DialogOK

Signatur

```
procedure DialogOK(D:PParameterStruct);
                                              export far pascal;

void far pascal DialogOK(PParameterStruct D)
```

Parameter *D* ist ein Zeiger auf die Struktur *TParameterStruct.*

Funktion *DialogOK* ist in der Regel nur bei User-Blöcken mit Visualisierungsfenster erforderlich und dort auch nur in wenigen Fällen. Diese Funktion wird von BORIS aufgerufen, wenn der Anwender den Parameterdialog des User-Blocks über den *OK*-Schalter verlassen hat. Sie kann beispielsweise benutzt werden, um den Inhalt des Visualisierungsfensters an die geänderten Blockparameter anzupassen.

Beispiel:

Folgendes Listing zeigt ein Beispiel für die Anwendung von *DialogOK*.

```
procedure DialogOK(D:PParameterStruct); export far pascal;
begin
  {Nachdem der Parameterdialog über OK verlassen wurde,
  soll das Visualisierungsfenster, dessen Handle in
  UserHWindow liegt, aufgefrischt werden!}
  InvalidateRect(D^.UserHWindow, nil, true);
end;
```

Beispiel für die Anwendung von DialogOK

ShowWindowDLL
HideWindowDLL

Signatur

```
Procedure ShowWindowDLL(D: PParameterStruct);
                                          export far pascal;
Procedure HideWindowDLL(D: PParameterStruct);
                                          export far pascal;

void far pascal ShowWindowDLL(PParameterStruct D)
void far pascal HideWindowDLL(PParameterStruct D)
```

Parameter *D* ist ein Zeiger auf die Struktur *TParameterStruct.*

Funktion Diese Funktionen werden nur bei User-DLLs mit Visualisierungsfenstern benötigt. Sie können benutzt werden, um das Visualisierungsfenster anzuzeigen bzw. zum Symbol zu verkleinern. Die Funktionen werden von BORIS aufgerufen, wenn der Anwender die Option OPTIONEN | ALLE ANZEIGEFENSTER ZEIGEN bzw. OPTIONEN | ALLE ANZEIGEFENSTER VERBERGEN anwählt.

Beispiel:

Folgendes Listing zeigt ein Anwendungsbeispiel für die beiden Funktionen:

```
...
procedure ShowWindowDLL(D: PParameterStruct); export far pascal;
begin
  {Anzeigefenster in Normalgröße anzeigen}
  ShowWindow(D^.UserHWindow, sw_Normal);
end;

procedure HideWindowDLL(D: PParameterStruct); export far pascal;
begin
  {Anzeigefenster zum Symbol verkleinern}
  ShowWindow(D^.UserHWindow, sw_Minimize);
end;
```

Beispiel für die Anwendung von ShowWindowDLL *bzw.* HideWindowDLL

SetInputChar
SetOutputChar

Signatur

```
Function SetInputChar: PChar; export far pascal;
Function SetOutputChar: PChar; export far pascal;

char far pascal SetInputChar(void)
char far pascal SetOutputChar(void)
```

Rückgabewert Der Rückgabewert enthält einen String, dessen Zeichen die Beschriftung der einzelnen Blockeingänge bzw. Blockausgänge festlegen.

Funktion *SetInputChar* bzw. *SetOutputChar* können benutzt werden, um die standardmäßige Beschriftung der Blockeingänge bzw. -ausgänge des User-Blocks in der Strukturdarstellung zu ändern. In der Regel wird der Eingang eines Blocks mit *E* (bei nur einem Eingang) und der Ausgang mit *A* (bei nur einem Ausgang) beschriftet, bei mehreren Ein- bzw. Ausgängen werden diese durchnumeriert. Durch Implementierung von *SetInputChar* bzw. *SetOutputChar* kann für jeden Ein- bzw. Ausgang stattdessen ein benutzerdefiniertes Zeichen vorgegeben werden.

Beispiel:

Bei einem User-Block mit zwei Eingängen und einem Ausgang sollen die Eingänge mit *w* und *x* und der Ausgang mit *y* beschriftet werden. Nachfolgendes Listing zeigt die entsprechende Implementierung der Funktionen *SetInputChar* und *SetOutputChar*.

```
Function SetInputChar: PChar; export far pascal;
begin
  SetInputChar := 'wx'; {Eingang 1 mit "w", Eingang 2 mit "x" beschriften}
end;

Function SetOutputChar: PChar; export far pascal;
begin
  SetOutputChar := 'y'; {Ausgang mit "y" beschriften}
end;
```

Beispiel für die Anwendung von SetInputChar *und* SetOutputChar

GetDLLName

Signatur

```
Function GetDLLName: PChar; export far pascal;

char far pascal GetDLLName(void)
```

Rückgabewert Bezeichnung des User-DLL-Blocks

Funktion Über diese Funktion kann dem User-DLL-Block ein Name gegeben werden, der dann im Untermenü SYSTEMBLÖCKE | USER-DLL-BLÖCKE bzw. im ToolTip-Fenster der Palettenseite *User* der Systemblock-Toolbar erscheint.

Beispiel:

Nachfolgend definierter User-DLL-Block erhält die Bezeichnung *Riesen-Display*.

```
function GetDLLName: PChar; export pascal;
begin
  GetDLLName := 'Riesen-Display';
end;
```

Beispiel für die Realisierung von GetDLLName

CallParameterDialogDLL

Signatur

```
Procedure CallParameterDialogDLL(D1: PParameterStruct;
            D2: PNumberOfInputsOutputs); export pascal;

void far pascal CallParameterDialogDLL
     (PParameterStruct D1, PNumberOfInputsOutputs D2)
```

Parameter *D1* ist ein Zeiger auf die Struktur *TParameterStruct.*

D2 ist ein Zeiger auf die Struktur *TNumberOfInputsOutputs.*

Funktion Diese Funktion ermöglicht den Aufruf eines benutzerdefiniertes Parameterdialogs anstelle des Standard-Dialogs.

Beispiel:

Nachfolgendes Listing aus der Beispiel-DLL DLLDEMO2 zeigt den Aufruf eines benutzerdefinierten Dialogs in DELPHI.

```
procedure CallParameterDialogDLL(D1:PParameterStruct;
                                 D2:PNumberOfInputsOutputs);export pascal;
(* Stellt den benutzerdefinierten Parameterdialog dar *)
var s: string;
    Code: integer;
begin
  Form1 := TForm1.Create(Application);
  Str(D1^.E[0]:10, s);
  Form1.Edit1.Text := s;
  Form1.Checkbox1.Checked := D1^.B[0] > 0;
  if Form1.ShowModal = mrOk then begin
    Val(Form1.Edit1.Text, D1^.E[0], Code);
    D1^.B[0] := ord(Form1.Checkbox1.Checked);
  end;
  Form1.Free;
end;
```

Beispiel für die Realisierung von CallParameterDialogDLL

WriteToFile
ReadFromFile
NumberOfLinesInSystemFile

Signatur

```
Procedure WriteToFile(AFileHandle: word;
            D: PParameterStruct); export far pascal
Procedure ReadFromFile(AFileHandle: word;
            D: PParameterStruct); export far pascal
Function  NumberOfLinesInSystemFile: word;
                                     export far pascal

void far pascal WriteToFile(int AFileHandle,
                            PParameterStruct D)
void far pascal ReadFromFile(int AFileHandle,
                            PParameterStruct D)

int far pascal NumberOfLinesInSystemFile(void);
```

Parameter *AFileHandle* ist das Handle der Systemdatei.

D ist ein Zeiger auf die Struktur *TParameterStruct.*

Funktion Diese drei Funktionen dienen zur Speicherung von blockspezifischen Parametern, die über die 32 Fließkomma-, Ganzzahl- und Schalterparameter hinausgehen. Sie sind daher nur in seltenen Fällen notwendig. Die User-DLL BIGDIS32.DLL im Verzeichnis *UserDLLs* demonstriert die Anwendung der drei Funktionen.

CanSimulateDLL

Signatur

```
function CanSimulateDLL(D:PParameterStruct):Integer;
                                          export far pascal;

int far pascal CanSimulateDLL(PParameterStruct D)
```

Parameter *D* ist ein Zeiger auf die Struktur *TParameterStruct.*

Rückgabewert Der Rückgabewert '0' zeigt an, daß nicht simuliert werden kann, der Rückgabewert von '1', daß dieser Block simulationsbereit ist.

Funktion *CanSimulateDLL* wird von BORIS vor Beginn der Simulation (noch vor *InitSimulationDLL*) aufgerufen, um zu überprüfen, ob der User-Block simuliert werden kann. Mittels dieser Funktion ist es also möglich, unter bestimmten Bedingungen (beispielsweise, wenn der Block eine Parameterdatei benötigt und für diese ein nicht existierender Dateiname angegeben wurde) eine Simulation zu verhindern, indem der Rückgabewert auf 0 gesetzt wird. In den meisten Fällen sind derartige Überprüfungen nicht notwendig und der Rückgabewert kann auf 1 gesetzt werden.

Beispiel:

Folgende Listings zeigen zwei Realisierungsmöglichkeiten für die Funktion *CanSimulateDLL*.

```
Function CanSimulateDLL(D: PParameterStruct): integer; export far pascal;
begin
  CanSimulateDLL := 1; {Simulation immer zulässig!}
end;
```

```
Function CanSimulateDLL(D: PParameterStruct): integer; export far pascal;
begin
  {Die Simulation soll nur zulässig sein, wenn Fließkomma-Parameter 0
  und Fließkomma-Parameter 1 unterschiedliches Vorzeichen haben!}
  if D^.E[0] * D^.E[1] < 0 then
    CanSimulateDLL := 1
  else
    CanSimulateDLL := 0;
end;
```

Beispiele für die Realisierung von CanSimulateDLL

InitSimulationDLL

Signatur

```
procedure InitSimulationDLL (D:PParameterStruct;
                             Inputs:PInputArray;
                             Outputs:POutputArray);
                             export far pascal;

void far pascal InitSimulationDLL (PParameterStruct D,
                                   TInputArray FAR Inputs,
                                   TOutputArray FAR Outputs)
```

Parameter *D* ist ein Zeiger auf die Struktur *TParameterStruct.*

Inputs ist ein Zeiger auf die Struktur *TInputArray.*

Outputs ist ein Zeiger auf die Struktur *TOutputArray*.

Funktion *InitSimulationDLL* initialisiert den Block für den Zeitpunkt $t = 0$ bzw. den ersten Simulationsschritt. Hier können also Voreinstellungen vorgenommen werden, die vor Beginn der eigentlichen Simulationsschleife notwendig werden (z. B. Setzen von Anfangswerten oder inneren Zustandsgrößen, Vorbelegung der Ausgangswerte usw.). Sofern erforderlich, können innerhalb dieser Funktion z. B. auch Daten, die für die Simulation benötigt werden, aus einer externen Parameterdatei gelesen werden (siehe Beispiel *Kennfeld-DLL* an späterer Stelle). Die aktuellen Eingangswerte des Blocks werden über den Zeiger *Inputs* übergeben; die berechneten Ausgangswerte müssen über den Zeiger *Outputs* zurückgeliefert werden.

Beispiel:

Nachfolgendes Listing zeigt eine beispielhafte Realisierung von *InitSimulationDLL* für einen Block mit zwei Eingängen und zwei Ausgängen. In diesem Fall soll zum Zeitpunkt $t = 0$ der erste Eingang auf den ersten Ausgang durchgeschaltet werden und der zweite Ausgang zu 0 initialisiert werden.

```
procedure InitSimulationDLL (D:PParameterStruct; Inputs:PInputArray;
                             Outputs:POutputArray); export far pascal;
begin
  Outputs^[1] := Inputs^[1];  {Ausgang 1 auf Eingang 1}
  Outputs^[2] := 0.0;         {Ausgang 2 zu 0 initialisieren}
end;
```

Beispiel für die Realisierung von InitSimulationDLL

SimulateDLL

Signatur

```
Procedure SimulateDLL(T:Extended; D:PParameterStruct;
                      Inputs:PInputArray;
                      Outputs:POutputArray);
                      export far pascal;

void far pascal SimulateDLL(long double T,
                            PParameterStruct D,
                            TInputArray FAR Inputs,
                            TOutputArray FAR Outputs)
```

Parameter *T* ist der Zeitwert des augenblicklichen Simulationsschrittes

D ist ein Zeiger auf die Struktur *TParameterStruct.*

Inputs ist ein Zeiger auf die Struktur *TInputArray*.

Outputs ist ein Zeiger auf die Struktur *TOutputArray*.

Funktion *SimulateDLL* enthält den eigentlichen numerischen Kern des User-Blocks, d. h. den zu jedem Simulationsschritt abzuarbeitenden Algorithmus. Diese Funktion wird zu jedem Simulationsschritt aufgerufen. Der Parameter *T* enthält dabei den momentanen Zeitwert, der Parameter *D* einen Zeiger auf die Blockparameter. Die aktuellen Eingangswerte des Blocks werden über den Zeiger *Inputs* übergeben; die berechneten Ausgangswerte müssen über den Zeiger *Outputs* zurückgeliefert werden.

Wird im Simulationsalgorithmus auch die Simulationsschrittweite ΔT benötigt, kann statt der Funktion *SimulateDLL* die Funktion *SimulateDLL2* (wird nachfolgend beschrieben) benutzt werden, der die Schrittweite als zusätzlicher Parameter übergeben wird.

Beispiel:

Folgendes Listing zeigt die Realisierung von *SimulateDLL* für einen Block mit zwei Eingängen und zwei Ausgängen. Am ersten Ausgang wird der arithmetische Mittelwert beider Eingänge - zusätzlich multipliziert mit Fließkomma-Parameter 0 - und am zweiten Ausgang der geometrische Mittelwert - multipliziert mit Fließkomma-Parameter 1 - ausgegeben.

```
Procedure SimulateDLL(T:Extended; D:PParameterStruct;
                      Inputs:PInputArray;
                      Outputs:POutputArray); export far pascal;
begin
  Outputs^[1] := D^.E[0] * (Inputs^[1] + Inputs^[2]) | 2;
  Outputs^[2] := D^.E[1] * sqrt((Inputs^[1] * Inputs^[2]));
end;
```

Beispiel für die Realisierung von SimulateDLL

EndSimulationDLL

Signatur

```
procedure EndSimulationDLL; export far pascal;
```

```
void far pascal EndSimulationDLL(void)
```

Funktion *EndSimulationDLL* wird aufgerufen, sobald die Simulation beendet wurde. In dieser Funktion können somit z. B. geöffnete Dateien geschlossen werden. In den meisten Anwendungen kann diese Funktion jedoch leer bleiben.

Statt *EndSimulationDLL* kann auch die Funktion *EndSimulationDLL2* (wird nachfolgend beschrieben) benutzt werden, der als zusätzlicher Parameter ein Zeiger auf die Struktur *TParameterStruct* übergeben wird.

Beispiel:

Nachfolgendes Listing zeigt ein Beispiel für die Anwendung von *EndSimulationDLL*.

```
procedure EndSimulationDLL; export far pascal;
begin
  Close(DataFile);   {Parameterdatei schließen}
end;
```

Beispiel für die Anwendung von EndSimulationDLL

SimulateDLL2

Signatur

```
Procedure SimulateDLL2(T, DeltaT:Extended;
                       D:PParameterStruct;
                       Inputs:PInputArray;
                       Outputs:POutputArray);
```



```
                                export far pascal;
void far pascal SimulateDLL2(long double T,
                             long double DeltaT,
                             PParameterStruct D,
                             TInputArray FAR Inputs,
                             TOutputArray FAR Outputs)
```

Parameter *T* ist der Zeitwert des augenblicklichen Simulationsschrittes

DeltaT ist die Simulationsschrittweite.

D ist ein Zeiger auf die Struktur *TParameterStruct.*

Inputs ist ein Zeiger auf die Struktur *TInputArray*.

Outputs ist ein Zeiger auf die Struktur *TOutputArray*.

Funktion *SimulateDLL2* kann statt der Funktion *SimulateDLL* benutzt werden, wenn zur Verarbeitung auch die Simulationsschrittweite ΔT benötigt wird.

EndSimulationDLL2

Signatur

```
procedure EndSimulationDLL2(D: PParameterStruct);
                                          export far pascal;
void far pascal EndSimulationDLL2(PParameterStruct D)
```

Parameter *D* ist ein Zeiger auf die Struktur *TParameterStruct.*

Funktion *EndSimulationDLL2* kann statt der Funktion *EndSimulationDLL* benutzt werden, wenn zur Verarbeitung auch die Parameterstruktur *TParameterStruct* benötigt wird.

SetEnhancedInformation

Signatur

```
procedure SetEnhancedInformation
                                (DeltaT, TSimu: extended;
                                 D: PParameterStruct);
                                 export far pascal;
void far pascal SetEnhancedInformation
                                  (long double DeltaT,
                                   long double TSimu,
                                   PParameterStruct D)
```

Parameter *DeltaT* ist die Simulationsschrittweite.

TSimu ist die Simulationsdauer.

D ist ein Zeiger auf die Struktur *TParameterStruct.*

Funktion *SetEnhancedInformation* wird nur in seltenen Fällen benötigt. Die

Funktion liefert dem User-Block die eingestellte Simulationsschrittweite und -dauer zur weiteren Verwendung. Beide Größen können z. B. in nicht benutzten Fließkomma-Parametern zur weiteren Nutzung abgelegt werden.

Beispiel:

In nachfolgendem Beispiel werden *DeltaT* und *TSimu* in den Fließkomma-Parametern 3 und 4 gespeichert.

```
procedure SetEnhancedInformation(DeltaT, TSimu: extended;
                                 D: PParameterStruct); export far pascal;
begin
  D^.E[3] := DeltaT;
  D^.E[4] := TSimu;
end;
```

Beispiel für die Anwendung von SetEnhancedInformation

Beispiele

Beispiel 1: Eine simple User-DLL

Wir wollen zunächst beginnen mit einem einfachen Beispiel einer User-DLL mit zwei Eingängen und einem Ausgang. Der Block soll in Abhängigkeit vom ersten Eingang (Steuereingang) x_1 den Wert des zweiten Eingangs (Dateneingang) x_2 mit unterschiedlichen Verstärkungenfaktoren k_1, k_2, k_3 multiplizieren und am Ausgang y ausgeben. Über einen Schalter-Parameter b_1 soll der Ausgang zusätzlich invertiert werden können:

- Falls x_1 negativ ist, soll x_2 mit k_1 multipliziert werden.
- Falls x_1 null ist, soll x_2 mit k_2 multipliziert werden.
- Falls x_1 positiv ist, soll x_2 mit k_3 multipliziert werden.
- Falls b_1 gesetzt ist, soll der Ausgang zusätzlich invertiert werden.

Nachfolgendes Listing zeigt die Realisierung der DLL in Pascal. Die fertig compilierte DLL befindet sich als DLLDEMO1.DLL im *UserDLLs*-Verzeichnis, der zugehörige Quelltext unter DLLDEMO1.DPR im Unterverzeichnis *Sources*. Alle wesentlichen Kommentare zur DLL sind im Quelltext vermerkt.

```
Library DLLDemo1;
(*************************************************)
(*                                               *)
```

```
(*       Einfache Demo-DLL für BORIS            *)
(*       (Beispiel 1 des Handbuchs)             *)
(*                                              *)
(************************************************)

uses WinProcs,
  SysUtils,
  WinTypes;

type
  PParameterStruct=^TParameterStruct;
  TParameterStruct=packed record
    NuE : Byte;                    {Anzahl reeller Zahlenwerte}
    NuI : Byte;                    {Anzahl ganzer Zahlenwerte}
    NuB : Byte;                    {Anzahl Schalter}
    E:Array[0..31] of Extended;    {reelle Zahlenwerte}
    I:Array[0..31] of Integer;     {ganze Zahlenwerte}
    B:Array[0..31] of Byte;        {Schalter}
    D:Array[0..255] of char;       {event. Dateiname für weitere Daten.}
    EMin:Array[0..31] of Extended; {untere Eingabegrenze}
    EMax:Array[0..31] of Extended; {obere Eingabegrenze}
    IMin:Array[0..31] of Integer;  {untere Eingabegrenze}
    IMax:Array[0..31] of Integer;  {obere Eingabegrenze}
    NaE : Array[0..31,0..40] of char;    {Namen der reellen Zahlenwerten}
    NaI : Array[0..31,0..40] of char;    {Namen der ganzen Zahlenwerten}
    NaB : Array[0..31,0..40] of char;    {Namen der Schalter}
    {den Rest der Variablen von TParameterStruct können wir uns sparen,
     da wir ihn nicht benötigen!}
  end;

  PDialogEnableStruct=^TDialogEnableStruct;
  TDialogEnablestruct=packed record
    AllowE: Longint;               { Soll die Eingabe eines Wertes   }
    AllowI: Longint;               { un-/zulässig sein so ist das Bit}
    AllowB: Longint;               { des Allow?-Feldes 0 bzw. 1}
    AllowD: Byte;
  end;

  PNumberOfInputsOutputs=^TNumberOfInputsOutputs;
  TNumberOfInputsOutputs=packed record
    Inputs :Byte;                  {Anzahl Eingänge}
    Outputs:Byte;                  {Anzahl Ausgänge}
    NameI : Array[0..49,0..40] of char;
    NameO : Array[0..49,0..40] of char;
  end;

  PInputArray = ^TInputArray;
  TInputArray = packed array[1..50] of extended;
  POutputArray = ^TOutputArray;
  TOutputArray = packed array[1..50] of extended;


procedure GetParameterStruct(D:PParameterStruct);export pascal;
begin
  D^.NuE:=3; {3 Fließkomma-Parameter}
  D^.NuI:=0; {0 Integer-Parameter}
  D^.NuB:=1; {1 Schalter-Parameter}
  {Namen der Parameter}
  StrCopy(D^.NaE[0], 'Verstärkung k1 für Steuereing. < 0');
  StrCopy(D^.NaE[1], 'Verstärkung k2 für Steuereing. = 0');
  StrCopy(D^.NaE[2], 'Verstärkung k3 für Steuereing. > 0');
  StrCopy(D^.NaB[0], 'Ausgang invertiert');
  {Parameter initialisieren}
  D^.E[0] := 1;
  D^.E[1] := 5;
  D^.E[2] := 10;
```



```
  D^.B[0] := 0;
  {Grenzwerte für Parameter festlegen}
  D^.EMin[0] := 0.0; D^.EMax[0] := 100000;
  D^.EMin[1] := 0.0; D^.EMax[1] := 100000;
  D^.EMin[2] := 0.0; D^.EMax[2] := 100000;
end;

procedure GetDialogEnableStruct(D:PDialogEnableStruct;
                                D2:PParameterStruct); export pascal;
begin
  {Alle Dialogelemente sollen jederzeit zugänglich sein}
  D^.AllowE := $FFFFFFFF;
  D^.AllowI := $FFFFFFFF;
  D^.AllowB := $FFFFFFFF;
end;

procedure GetNumberOfInputsOutputs(D:PNumberOfInputsOutputs);
                                                    export pascal;
begin
  D^.Inputs:=2;           { Der Block soll zwei Eingänge und}
  D^.Outputs:=1;          {           einen Ausgang haben !}
  StrPCopy(D^.NameI[0],'Dateneingang');    {Namensgebung des 1. Eingangs}
  StrPCopy(D^.NameI[1],'Steuereingang');   {Namensgebung des 2. Eingangs}
  StrPCopy(D^.NameO[0],'Ausgang');         {Namensgebung des 1. Ausgangs}
end;

function CanSimulateDLL(D:PParameterStruct):Integer; export pascal;
begin
  {Simulation immer zulässig!}
  CanSimulateDLL := 1;
end;

procedure SimulateDLL(T:Extended;D:PParameterStruct;Inputs:PInputArray;
                      Outputs:POutputArray);export pascal;
begin
  if Inputs^[1] < 0 then {Eingang 1 negativ}
    Outputs^[1] := D^.E[0] * Inputs^[2]
  else if Inputs^[1] = 0 then {Eingang 1 null}
    Outputs^[1] := D^.E[1] * Inputs^[2]
  else {Eingang 1 positiv}
    Outputs^[1] := D^.E[2] * Inputs^[2];
  if D^.B[0] = 1 then {Schalter-Parameter 0 = 1 ==> Ausgang invertieren}
    Outputs^[1] := -Outputs^[1];
end;

procedure InitSimulationDLL(D:PParameterStruct;Inputs:PInputArray;
                            Outputs:POutputArray);export pascal;
begin
  {Der Initialisierungsschritt soll genauso durchgeführt werden wie alle
   Simulationsschritte. Also rufen wir einfach SimulateDLL auf!}
  SimulateDLL(0, D, Inputs, Outputs);
end;

procedure EndSimulationDLL; export pascal;
begin
  {wird nicht benötigt}
end;

function SetInputChar: PChar; export pascal;
begin
  {Steuereingang mit 'S', Dateneingang mit 'D' beschriften}
  SetInputChar := 'SD';
end;

procedure IsUserDLL32; export pascal;
begin
```

```
end;

{Exportieren der notwendigen Funktionen und Prozeduren }
exports
  GetParameterStruct,
  GetDialogEnableStruct,
  GetNumberOfInputsOutputs,
  CanSimulateDLL,
  InitSimulationDLL,
  SimulateDLL,
  EndSimulationDLL,
  SetInputChar,
  IsUserDLL32;
begin
  {Initialisierung der DLL (nicht notwendig)}
end.
```

Pascal-Listing zu Beispiel 1

Beispiel 2: Benutzerdefiniertes Kennfeld

Als Erweiterung zur benutzerdefinierten Kennlinie, die in der BORIS-Systemblockbibliothek ja standardmäßig verfügbar ist, soll eine User-DLL zur Realisierung eines benutzerdefinierten *Kennfelds* der Form $z = f(x, y)$ erstellt werden. Die User-DLL soll folgende Spezifikationen erfüllen:

- Das Kennfeld soll in Matrixform als FWM-Datei (siehe Abschnitt *WinFACT-Dateitypen* in Kapitel *2 Grundlagen*) eingelesen werden. Der Name der Datei soll über den Parameterdialog vorgebbar sein. Der zugrundeliegende Abszissenbereich $[\, x_{\min,} x_{\max}\,]$ bzw. $[\, y_{\min,} y_{\max}\,]$ soll ebenfalls über den Parameterdialog des User-Blocks einstellbar sein. Die Matrix soll maximal die Dimension 20×20 aufweisen können.
- Zwischen den Stützpunkten soll auf Wunsch linear interpoliert werden können. Liegt ein Eingangswertepaar (x, y) außerhalb des von der Stützstellenmatrix erfaßten Bereichs, soll extrapoliert werden.
- Neben dem eigentlichen *z*-Ausgang soll ein zweiter Ausgang anzeigen, ob ein Eingangswertepaar (x, y) außerhalb des von der Stützstellenmatrix erfaßten Bereichs liegt, also extrapoliert wird. In diesem Fall soll dieser Ausgang High-Pegel, sonst Low-Pegel aufweisen. Die Werte für Low- und High-Pegel sollen ebenfalls über den Parameterdialog einstellbar sein.

Nachfolgendes Listing zeigt die Realisierung in Pascal. Die fertig compilierte DLL befindet sich als DRDFELD.DLL im *UserDLLs*-Verzeichnis, der zugehörige Quelltext unter DRDFELD.DPR im Unterverzeichnis *Sources*. Alle wesentlichen Kommentare zur DLL sind im Quelltext vermerkt.

```
Library DRDFeld;

(*********************************************)
(*                                           *)
(*         Kennfeld-User-DLL für BORIS       *)
(*          (Beispiel 2 des Handbuchs)       *)
(*                                           *)
(*********************************************)

uses Windows, SysUtils;

type
  PParameterStruct=^TParameterStruct;
  TParameterStruct= packed record
    NuE : Byte;                     {Anzahl reeller Zahlenwerte}
    NuI : Byte;                     {Anzahl ganzer Zahlenwerte}
    NuB : Byte;                     {Anzahl Schalter}
    E:Array[0..31] of Extended;     {reelle Zahlenwerte}
    I:Array[0..31] of Integer;      {ganze Zahlenwerte}
    B:Array[0..31] of Byte;         {Schalter}
    D:Array[0..255] of char;        {event. Dateiname für weitere Daten.}
    EMin:Array[0..31] of Extended;
    EMax:Array[0..31] of Extended;
    IMin:Array[0..31] of Integer;
    IMax:Array[0..31] of Integer;
    NaE : Array[0..31,0..40] of char;
    NaI : Array[0..31,0..40] of char;
    NaB : Array[0..31,0..40] of char;
    UserDataPtr: Pointer;
    {Den Rest von TParameterStruct können wir uns sparen, da wir ihn
     nicht benötigen!}
  end;

  PDialogEnableStruct=^TDialogEnableStruct;
  TDialogEnablestruct=packed record
    AllowE: Longint;                { Soll die Eingabe eines Wertes   }
    AllowI: Longint;                { un-/zulässig sein so ist das Bit}
    AllowB: Longint;                { des Allow?-Feldes 0 bzw. 1}
    AllowD: Byte;
  end;

  PNumberOfInputsOutputs=^TNumberOfInputsOutputs;
  TNumberOfInputsOutputs=packed record
    Inputs :Byte;                       {Anzahl Eingänge}
    Outputs:Byte;                       {Anzahl Ausgänge}
    NameI : Array[0..49,0..40] of char;
    NameO : Array[0..49,0..40] of char;
  end;

  PInputArray = ^TInputArray;
  TInputArray = packed array[1..50] of extended;
  POutputArray = ^TOutputArray;
  TOutputArray = packed array[1..50] of extended;


  {TUserData enthält die Kennfelddaten}
  TSingleMatrix = Array[1..20, 1..20] of Single;
  PUserData = ^TUserData;
  TUserData = record
                z: TSingleMatrix; {Funktionswertmatrix}
                nx,ny: Word;      {Anzahl Spalten bzw. Zeilen der Matrix}
                dx,dy: Extended;  {x- bzw. y-Abstand zwischen zwei
                                   Stützpunkten}
              end;

Const
```

```
  {Zwei Konstanten-Deklarationen für später...}
  SingleMin=-3.4E38;
  SingleMax= 3.4E38;


procedure GetParameterStruct(D:PParameterStruct);export pascal;
begin
  D^.NuE:=6;               {Sechs Fließpunktparameter}
  D^.NuI:=0;               {Keine Ganzzahlparameter }
  D^.NuB:=1;               {Ein Schalter}
  StrPCopy(D^.D,'*.fwm');  {Dateien mit Endung fwm zulassen}
  {Initialisierung der verwendeten Daten}
  D^.B[0]:=0;
  D^.E[0]:=-1; D^.EMin[0]:=SingleMin; D^.EMax[0]:=SingleMax;
  D^.E[1]:=1;  D^.EMin[1]:=SingleMin; D^.EMax[1]:=SingleMax;
  D^.E[2]:=-1; D^.EMin[2]:=SingleMin; D^.EMax[2]:=SingleMax;
  D^.E[3]:=1;  D^.EMin[3]:=SingleMin; D^.EMax[3]:=SingleMax;
  D^.E[4]:=0;  D^.EMin[4]:=SingleMin; D^.EMax[4]:=SingleMax;
  D^.E[5]:=5;  D^.EMin[5]:=SingleMin; D^.EMax[5]:=SingleMax;
  {Namensgebung max. 40 Zeichen !}
  StrPCopy(D^.NaE[0],'x - Achsenanfang');
  StrPCopy(D^.NaE[1],'x - Achsenende');
  StrPCopy(D^.NaE[2],'y - Achsenanfang');
  StrPCopy(D^.NaE[3],'y - Achsenende');
  StrPCopy(D^.NaE[4],'Low-Pegel für Ausgang 2');
  StrPCopy(D^.NaE[5],'High-Pegel für Ausgang 2');
  StrPCopy(D^.NaB[0],'lineare Interpolation');
end;

procedure GetDialogEnableStruct(D:PDialogEnableStruct;
                                D2:PParameterStruct); export pascal;
begin
  {Alle Dialogelemente sollen jederzeit zugänglich sein!}
  D^.AllowE:=$FFFFFFFF;
  D^.AllowB:=$FFFFFFFF;
  D^.AllowI:=$FFFFFFFF;
  D^.AllowD:= 1;
end;

procedure GetNumberOfInputsOutputs(D:PNumberOfInputsOutputs);export pascal;
begin
  D^.Inputs:=2;            { Der Block soll zwei Eingänge und}
  D^.Outputs:=2;           {                zwei Ausgänge haben !}
  StrPCopy(D^.NameI[0],'Abszissen-Wert (x)');    {Name des 1. Eingangs}
  StrPCopy(D^.NameI[1],'Abszissen-Wert (y)');    {Name des 2. Eingangs}
  StrPCopy(D^.NameO[0],'Ordinate (z) ');         {Name des 1. Ausgangs}
  StrPCopy(D^.NameO[1],'Extrapolation ON/OFF');  {Name des 2. Ausgangs}
end;

function CanSimulateDLL(D:PParameterStruct):Integer; export pascal;
var Datei: Text;
    Dateiname : Array[0..255] of char;
begin
  {Simulation nur zulassen, falls die angegebene FWM-Datei auch existiert!}
  StrCopy(Dateiname, D^.D);
  if Pos('*',StrPas(Dateiname))=0 then begin
    Assign(Datei,Dateiname);
    {$I-} Reset(Datei); {$I+}
    if IOResult = 0 then begin
      CanSimulateDLL := 1;
      Close(Datei);
    end else CanSimulateDLL := 0;
  end else CanSimulateDLL:=0;
end;

procedure SimulateDLL(T:Extended;D:PParameterStruct;
```



```
                    Inputs:PInputArray;Outputs:POutputArray);export pascal;
var ix, iy: integer;
    x0, y0, xp, yp: Extended;
    Extrapolation, Interpolation: boolean;
begin
  with PUserData(D^.UserDataPtr)^ do begin
    Extrapolation := (Inputs^[1]<D^.E[0]) or (Inputs^[1]>D^.E[1]) or
                     (Inputs^[2]<D^.E[2]) or (Inputs^[2]>D^.E[3]);
    if Extrapolation then
      Outputs^[2]:=D^.E[5]   {Ausgang 2 auf High-Pegel}
    else
      Outputs^[2]:=D^.E[4];  {Ausgang 2 auf Low-Pegel}
    Interpolation := D^.B[0] = 1;
    {Indizes ix, iy des nächstgelegenen Stützpunkts bestimmen}
    ix := trunc((Inputs^[1] - D^.E[0])/dx) + 1;
    iy := trunc((Inputs^[2] - D^.E[2])/dy) + 1;
    if ix < 1 then ix := 1;
    if iy < 1 then iy := 1;
    if Interpolation then begin
      if ix > nx-1 then ix := nx-1;
      if iy > ny-1 then iy := ny-1;
      x0 := D^.E[0] + (ix-1)*dx; {Referenzpunkt, x-Koordinate}
      y0 := D^.E[2] + (iy-1)*dy; {Referenzpunkt, y-Koordinate}
      xp := (z[ix+1, iy] - z[ix, iy]) / dx; {Steigung in x-Richtung}
      yp := (z[ix, iy+1] - z[ix, iy]) / dy; {Steigung in y-Richtung}
      Outputs^[1] := z[ix, iy] + (Inputs^[1] - x0)*xp + (Inputs^[2]-y0)*yp;
    end else begin
      if ix > nx then ix := nx;
      if iy > ny then iy := ny;
      Outputs^[1] := z[ix, iy];
    end;
  end;
end;

procedure InitSimulationDLL(D:PParameterStruct;Inputs:PInputArray;
                            Outputs:POutputArray);export pascal;
var i, j: Integer;
    Datei: text;
begin
  {Erst Parameterdatei einlesen...}
  assign(Datei, D^.D);
  reset(Datei);
  with PUserData(D^.UserDataPtr)^ do begin
    readln(Datei,ny,nx);
    for i:=1 to ny do
      for j:=1 to nx do readln(Datei,z[j, i]);
    dx:=(D^.E[1]-D^.E[0])/(nx-1); {Segmentgröße in x-Richtung errechnen!}
    dy:=(D^.E[3]-D^.E[2])/(ny-1); {Segmentgröße in y-Richtung errechnen!}
  end;
  close(Datei);
  {...und dann einen normalen Simulationsschritt anschließen}
  SimulateDLL(0, D, Inputs, Outputs);
end;

procedure EndSimulationDLL;export pascal;
begin
  {wird nicht benötigt}
end;

procedure InitUserData(D: PParameterStruct); export pascal;
begin
  {Speicherplatz für User-Daten anlegen}
  GetMem(D^.UserDataPtr, sizeof(TUserData));
end;

procedure DisposeUserData(D: PParameterStruct); export pascal;
```

```
begin
  {Speicherplatz für User-Daten wieder freigeben}
  FreeMem(D^.UserDataPtr, sizeof(TUserData));
end;

function SetInputChar: PChar; export pascal;
begin
  {Ersten Eingang mit 'x', zweiten mit 'y' beschriften}
  SetInputChar := 'xy';
end;

function SetOutputChar: PChar; export pascal;
begin
  {Ersten Ausgang mit 'z', zweiten mit 'E' beschriften}
  SetOutputChar := 'zE';
end;

procedure IsUserDLL32; export pascal;
begin
end;

{Exportieren der notwendigen Funktionen und Prozeduren }
exports
  GetParameterStruct, GetDialogEnableStruct, GetNumberOfInputsOutputs,
  CanSimulateDLL, InitSimulationDLL, SimulateDLL,
  InitUserData, DisposeUserData, EndSimulationDLL,
  SetInputChar, SetOutputChar, IsUserDLL32;

begin
  {Weitere Initialisierung der DLL (nicht notwendig).}
end.
```

Pascal-Listing für Beispiel 2

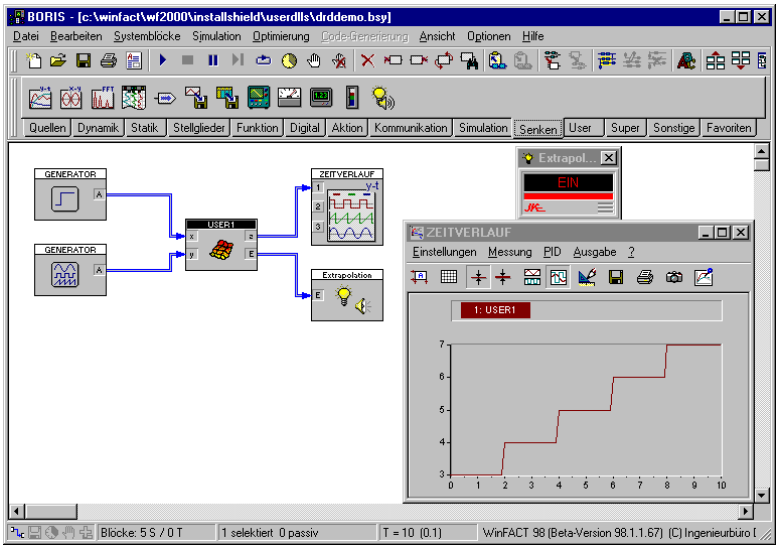
Anwendungsbeispiel für Kennfeld-DLL

Es soll ein Kennfeld definiert werden für den Bereich $0 \le x \le 4$, $0 \le y \le 4$, bei dem der Ausgangswert (z-Achse) linear von 1 bis 9 ansteigt. Das Kennfeld soll in beiden Richtungen über jeweils 5 Stützpunkte festgelegt sein. Dann hat die zugehörige Kennfeldmatrix folgendes Aussehen:

$$\underline{M} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{pmatrix}.$$

Die entsprechende FWM-Datei befindet sich unter dem Namen DRDDE-MO.FWM im Verzeichnis *UserDLLs*. Es soll der Verlauf der Ausgangsgröße für den Fall simuliert werden, daß die Eingangsgröße x konstant 2 ist und die Eingangsgröße y linear von 0 auf 5 ansteigt. Es soll dabei keine Interpolation stattfinden. Nachfolgende Grafik zeigt das entsprechende Simulationsergebnis.

Die zugehörige Systemdatei befindet sich unter dem Namen DRDDEMO.BSY ebenfalls im Verzeichnis *UserDLLs.*

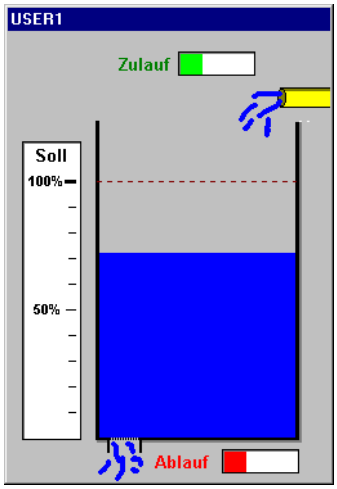


Simulationsergebnis

Beispiel 3: Visualisierung einer Füllstandsregelung

Es soll eine User-DLL erstellt werden, die zur Visualisierung einer Füllstandsregelung dient. Dabei soll beim Starten des ersten Simulationslaufs ein Ausgabefenster erscheinen, in dem die Eingangsgrößen des Blocks (Zulauf, Ablauf, Füllhöhe) visualisiert werden (siehe nachfolgende Grafik). Dieses Ausgabefenster soll erst dann wieder verschwinden, wenn der Block gelöscht wird. Bei der Programmierung soll davon ausgegangen werden, daß alle Eingangsgrößen zwischen 0 und 1 liegen (entsprechend 0 bzw. 100% Zulauf, Ablauf oder Füllhöhe).

Nachfolgendes Listing zeigt die Realisierung der DLL allein auf Basis der Windows-API. Die fertige DLL befindet sich unter dem Namen FUELLSTD.DLL im Verzeichnis *UserDLLs*, der Quelltext FUELLSTD.DPR im Unterverzeichnis *Sources*. Eine wesentlich einfachere Programmierung derartiger Blocktypen läßt sich mit Entwicklungsumgebungen wie DELPHI, VISUAL BASIC etc. erreichen; eine ganze Reihe von in DELPHI 3 realisierten Beispielen (teilweise mit Quelltexten) finden Sie im *UserDLLs*- bzw. *Sources*-Verzeichnis.



Visualisierungsfenster der User-DLL

```
Library FuellStd;

(***********************************************)
(*                                             *)
(*        DLL zur Visualisierung einer         *)
(*            Füllstandsregelung               *)
(*         (Beispiel 3 des Handbuchs)          *)
(*                                             *)
(***********************************************)

uses WinProcs, WinTypes, Windows, SysUtils, Messages;

{$R FUELLBMP}  {RES-Datei Hintergrund-Bitmap für das Ausgabefenster}


const
  cxBitmap = 250;  (* Breite des Bitmaps *)
  cyBitmap = 350;  (* Höhe des Bitmaps   *)

 {Damit ggfls. auch mehrere DLL-Blöcke dieses Typs gleichzeitig simuliert
  werden können, müssen wir uns für jedes Fenster das Fensterhandle und die
  aktuellen Eingangswerte merken. Außerdem soll jedes Fenster den Namen des
  zugehörigen User-DLL-Blocks im Titel anzeigen. Eleganterweise macht man
  das in einer verketteten Liste. Damit es nicht ganz so kompliziert wird,
  wählen wir hier aber ein statisches Array der Dimension 100 (mehr Blöcke
  wird wohl kaum jemand nehmen!)}
var
  nWindows: integer;  {enthält die Anzahl der geöffneten Fenster}
  LevelWindows: array[1..100] of record
    HW: HWnd;                       {Fensterhandle}
    VPos, VNeg, Height: extended; {aktuelle Eingangswerte des Blocks:
                                    Zulauf, Ablauf, Füllstand}
  end;

type
  PParameterStruct=^TParameterStruct;
  TParameterStruct=packed record
    NuE : Byte;                       {Anzahl reeller Zahlenwerte}
```



```
    NuI : Byte;                       {Anzahl ganzer Zahlenwerte}
    NuB : Byte;                       {Anzahl Schalter}
    E:Array[0..31] of Extended;       {reelle Zahlenwerte}
    I:Array[0..31] of Integer;        {ganze Zahlenwerte}
    B:Array[0..31] of Byte;           {Schalter}
    D:Array[0..255] of char;          {event. Dateiname für weitere Daten.}
    EMin:Array[0..31] of Extended;
    EMax:Array[0..31] of Extended;
    IMin:Array[0..31] of Integer;
    IMax:Array[0..31] of Integer;
    NaE : Array[0..31,0..40] of char; {Namen der rellen Zahlenwerte}
    NaI : Array[0..31,0..40] of char; {Namen der ganzen Zahlenwerte}
    NaB : Array[0..31,0..40] of char; {Namen der Schalter}
    UserDataPtr: Pointer;             {benutzerdef. Daten (nicht benötigt)}
    ParentPtr: Pointer;               {Zeiger auf User-DLL-Block}
    ParentHWnd: HWnd;                 {BORIS-Fensterhandle}
    ParentName: PChar;                {Name des User-DLL-Blocks}
    UserHWindow: HWnd;                {Handle des Ausgabefensters}
    DataFile: text;                   {Text-Datei (hier nicht benötigt)}
  end;

  PDialogEnableStruct=^TDialogEnableStruct;
  TDialogEnablestruct=packed record
    AllowE: Longint;                { Soll die Eingabe eines Wertes   }
    AllowI: Longint;                { un-/zulässig sein so ist das Bit}
    AllowB: Longint;                { des Allow?-Feldes 0 bzw. 1}
    AllowD: Byte;
  end;

  PNumberOfInputsOutputs=^TNumberOfInputsOutputs;
  TNumberOfInputsOutputs=packed record
    Inputs :Byte;                         {Anzahl Eingänge}
    Outputs:Byte;                         {Anzahl Ausgänge}
    NameI : Array[0..49,0..40] of char;
    NameO : Array[0..49,0..40] of char;
  end;

  PInputArray = ^TInputArray;
  TInputArray = packed array[1..50] of extended;
  POutputArray = ^TOutputArray;
  TOutputArray = packed array[1..50] of extended;


function GetVPos(H: HWnd): extended;
{Diese Funktion liefert für das Fenster mit dem Handle H den zugehörigen
 Parameter VPos, d. h. den Zulauf zurück}
var i: integer;
begin
  {Fensterliste nach passendem Fenster durchsuchen}
  for i:=1 to nWindows do if LevelWindows[i].HW = H then
    GetVPos := LevelWindows[i].VPos;
end;

function GetVNeg(H: HWnd): extended;
{Diese Funktion liefert für das Fenster mit dem Handle H den zugehörigen
 Parameter VNeg, d. h. den Zulauf zurück}
var i: integer;
begin
  {Fensterliste nach passendem Fenster durchsuchen}
  for i:=1 to nWindows do if LevelWindows[i].HW = H then
    GetVNeg := LevelWindows[i].VNeg;
end;

function GetHeight(H: HWnd): extended;
{Diese Funktion liefert für das Fenster mit dem Handle H den zugehörigen
 Parameter Height, d. h. die Füllhöhe zurück}
```



```
var i: integer;
begin
  {Fensterliste nach passendem Fenster durchsuchen}
  for i:=1 to nWindows do if LevelWindows[i].HW = H then
    GetHeight := LevelWindows[i].Height;
end;

procedure SetInputs(H: HWnd; VP, VN, Hgt: extended);
{Setzt in der Fensterliste die Eingangswerte des Fensters mit Handle H}
var i: integer;
begin
  for i:=1 to nWindows do if LevelWindows[i].HW = H then begin
    LevelWindows[i].VPos := VP;     {Zulauf}
    LevelWindows[i].VNeg := VN;     {Ablauf}
    LevelWindows[i].Height := Hgt;  {Füllhöhe}
  end;
end;

procedure Paint(DC: HDC; VP, VN, Hgt: extended);
(* Übernimmt die eigentliche Zeichnung des Ausgabefensters *)
var MemDC: HDC;
    RedBrush, BlueBrush, GreenBrush: HBrush;
    HBM: HBitmap;
    BlockBitmap: TBitmap;
    Rect: TRect;
begin
  MemDC := CreateCompatibleDC(DC);
  HBM := LoadBitmap(HInstance, 'BEHAELTER_BITMAP');
  SelectObject(MemDC, HBM);
  GetObject(HBM, SizeOf(BlockBitmap), @BlockBitmap);
  RedBrush := CreateSolidBrush(RGB(255, 0, 0));
  BlueBrush := CreateSolidBrush(RGB(0, 0, 255));
  GreenBrush := CreateSolidBrush(RGB(0, 255, 0));
  {Zulauf}
  with Rect do begin
    Left := 134;
    Right := Left + round(VP*57);
    Top := 17;
    Bottom := 33;
    FillRect(MemDC, Rect, GreenBrush);
  end;
  {Ablauf}
  with Rect do begin
    Left := 168;
    Right := Left + round(VN*57);
    Top := 327;
    Bottom := 343;
    FillRect(MemDC, Rect, RedBrush);
  end;
  {Füllstand}
  with Rect do begin
    Left := 71;
    Right := 223;
    Bottom := 316;
    Top := Bottom - round(Hgt*200);
    FillRect(MemDC, Rect, BlueBrush);
  end;
  BitBlt(DC, 0, 0, BlockBitmap.bmwidth, BlockBitmap.bmheight, MemDC, 0, 0,
         SrcCopy);
  DeleteObject(RedBrush);
  DeleteObject(GreenBrush);
  DeleteObject(BlueBrush);
  DeleteDC(MemDC);
  DeleteObject(HBM);
end;
```



```
function WndFunc(Wnd: HWnd; Msg, wParam: word; lParam: longint): longint;
                                                                    stdcall;
{Fensterfunktion des Ausgabefensters}
var PaintStruct: TPaintStruct;
    DC: HDC;
    i, Index: integer;
begin
  case Msg of
    wm_Paint: begin
                DC := BeginPaint(Wnd, PaintStruct);
                Paint(DC, GetVPos(Wnd), GetVNeg(Wnd), GetHeight(Wnd));
                EndPaint(Wnd, PaintStruct);
              end;
    wm_Destroy: begin
                  {Das Fenster soll zerstört werden. Wir müssen es also
                   aus der Fensterliste löschen und alle anderen Einträge
                   um eins nach vorn verschieben}
                  Index := 0;
                  repeat inc(Index) until LevelWindows[Index].HW = Wnd;
                  for i:=Index+1 to nWindows do
                    LevelWindows[i-1] := LevelWindows[i];
                  dec(nWindows);
                end;
  end;
  WndFunc := DefWindowProc(Wnd, Msg, wParam, lParam);
end;

const
  WindowClass: TWndClass = (
    style: 0;
    lpfnWndProc: @WndFunc;
    cbClsExtra: 0;
    cbWndExtra: 0;
    hInstance: 0;
    hIcon: 0;
    hCursor: 0;
    hbrBackground: 1;
    lpszMenuName: nil;
    lpszClassName: 'LevelWindow');

procedure GetParameterStruct(D:PParameterStruct); export pascal;
begin
  (* keine Parameter erforderlich ! *)
  D^.NuE := 0;
  D^.NuI := 0;
  D^.NuB := 0;
end;

procedure GetDialogEnableStruct(D:PDialogEnableStruct;D2:PParameterStruct);
                                                              export pascal;
begin
  (* wird hier nicht benötigt, da kein Dialog erforderlich *)
end;

procedure GetNumberOfInputsOutputs(D:PNumberOfInputsOutputs);export pascal;
begin
  D^.Inputs:=3;    {Unser Block hat drei Eingänge...}
  D^.Outputs:=0;   {            und keinen Ausgang  }
  {Namen der iIngänge}
  StrCopy(D^.NameI[0], 'Zulauf');
  StrCopy(D^.NameI[1], 'Ablauf');
  StrCopy(D^.NameI[2], 'Füllstand');
end;

procedure InitUserDLL(D:PParameterStruct); export pascal;
{Diese Prozedur wird vom User-DLL-Block aufgerufen, wenn er initialisiert
```

```
 wird. Wir benötigen ein Flag, an dem wir später erkennen können,
 ob das Ausgabefenster schon existiert. Dazu nehmen wir den ersten
 Integer-Parameter D^.I[0]. Im Prinzip könnten wir das Ausgabefenster auch
 schon hier erzeugen; wir wollen aber, daß es erst bei Start der ersten
 Simulation erscheint; dadurch wird die ganze Sache etwas komplizierter!}
begin
  D^.I[0] := 0;
end;

procedure DisposeUserDLL(D:PParameterStruct); export pascal;
{Diese Prozedur wird vom User-DLL-Block aufgerufen, wenn der Block gelöscht
 wird oder eine andere DLL für den Block geladen wird. In diesen Fällen
 müssen wir das Ausgabefenster löschen}
begin
  DestroyWindow(D^.UserHWindow);
end;

function CanSimulateDLL(D:PParameterStruct):Integer; export pascal;
begin
  CanSimulateDLL:=1;  {Simulation immer möglich}
end;

procedure InitSimulationDLL(D:PParameterStruct;Inputs:PInputArray;
                            Outputs:POutputArray); export pascal;
var Rect: TRect;
    x, y: integer;
begin
  with Rect do begin
    Top := 0;
    Bottom := cyBitmap;
    Left := 0;
    Right := cxBitmap;
  end;
  (* Fenstergröße an Bitmapgröße anpassen! *)
  AdjustWindowRect(Rect, ws_Popup or ws_Caption, false);
  {Falls noch kein Ausgabefenster existiert, wird es jetzt angelegt...}
  if D^.I[0] = 0 then begin
    D^.I[0] := 1; {kennzeichnen, daß Fenster jetzt vorhanden}
    {Damit bei mehreren Fenster nicht alle aufeinander liegen, verschieben
     wir jedes um 50 Pixel nach links oben}
    x := 350 + nWindows*50;
    y := 70 - nWindows*50;
    {Es ist soweit: Das Fenster wird erzeugt...}
    D^.UserHWindow := CreateWindowEx(ws_ex_TopMost, 'LevelWindow', '',
                              ws_Popup or ws_Caption or ws_MinimizeBox,
                              x, y, Rect.Right-Rect.Left, Rect.Bottom-
                              Rect.Top, D^.ParentHWnd, 0, HInstance, nil);
    {...angezeigt...}
    ShowWindow(D^.UserHWindow, sw_Show);
    {...und in die Fensterliste eingetragen!}
    inc(nWindows);
    LevelWindows[nWindows].HW := D^.UserHWindow;
  end;
  {Wir wollen dem Fenster den Namen des User-DLL-Blocks geben}
  SetWindowText(D^.UserHWindow, D^.ParentName);
end;

procedure SimulateDLL(T:Extended;D:PParameterStruct;Inputs:PInputArray;
                      Outputs:POutputArray);export pascal;
var DC: HDC;
begin
  {Inputs in Fensterliste eintragen...}
  SetInputs(D^.UserHWindow, Inputs^[1], Inputs^[2], Inputs^[3]);
  {...und Fensterinhalt neu zeichnen}
  DC := GetDC(D^.UserHWindow);
  Paint(DC, GetVPos(D^.UserHWindow), GetVNeg(D^.UserHWindow),
```



```
       GetHeight(D^.UserHWindow));
  ReleaseDC(D^.UserHWindow, DC);
end;

procedure EndSimulationDLL; export pascal;
begin
  (* hier nicht erforderlich *)
end;

function SetInputChar: PChar; export pascal;
begin
  {Ersten Eingang mit '+', zweiten mit '-' und dritten mit 'H' beschriften}
  SetInputChar := '+-H';
end;

procedure ShowWindowDLL(D: PParameterStruct); export pascal;
begin
  {Anzeigefenster in Normalgröße anzeigen}
  ShowWindow(D^.UserHWindow, sw_Normal);
end;

procedure HideWindowDLL(D: PParameterStruct); export pascal;
begin
  {Anzeigefenster zum Symbol verkleinern}
  ShowWindow(D^.UserHWindow, sw_Minimize);
end;

procedure IsUserDLL32; export pascal;
begin
end;

exports
  InitUserDLL, DisposeUserDLL, GetParameterStruct,
  GetDialogEnableStruct, GetNumberOfInputsOutputs,
  CanSimulateDLL, InitSimulationDLL, SimulateDLL,
  EndSimulationDLL, SetInputChar, ShowWindowDLL,
  HideWindowDLL, IsUserDLL32;

begin
  (* Fensterklasse registrieren *)
  WindowClass.hInstance := HInstance;
  WindowClass.hIcon := LoadIcon(0, idi_Application);
  WindowClass.hCursor := LoadCursor(0, idc_Arrow);
  RegisterClass(WindowClass);
  nWindows := 0;
end.
```

Pascal-Listing zu Beispiel 3

Weitere Beispiele

Im Lieferumfang von BORIS befindet sich eine Vielzahl weiterer User-DLLs, die in der Regel in Pascal (DELPHI 3) entwickelt wurden. Die fertigen DLLs finden Sie im Unterverzeichnis *UserDLLs*, die zugehörigen Quelltexte - sofern vorhanden - im Unterverzeichnis *UserDLLs\Sources*. Einige dieser User-DLLs lassen sich direkt über die Palettenseite *User* der Systemblock-Toolbar erreichen.

Palettenseite User *der Systemblock-Toolbar*

Benutzerdefinierte Block-Bitmaps

Optional hat der Anwender die Möglichkeit, jeden seiner User-Blöcke mit einem *anwenderspezifischen Block-Bitmap* zu versehen (siehe auch Abschnitt *Arbeiten mit Superblöcken*). Dieses Block-Bitmap wird als BMP-Datei erstellt und muß den selben Namen aufweisen wie die DLL, jedoch mit der Extension BMP, und sich im selben Verzeichnis befinden wie die DLL. Zu einer DLL mit dem Namen MOTOR.DLL gehört also die Bitmap-Datei MOTOR.BMP. BORIS überprüft beim Einfügen eines User-Blocks automatisch, ob die zugehörige BMP-Datei vorhanden ist. Ist dies der Fall, wird diese benutzt, ansonsten das Standard-Userblock-Bitmap. Das Bitmap sollte eine Größe von 48×42 Pixeln aufweisen; Bitmaps anderer Größe werden automatisch gedehnt bzw. gestaucht, so daß sie optimal in das Blockschaltbild passen. Auch für die Druckerausgabe kann ein benutzerdefiniertes Bitmap (möglichst als s/w-Bitmap) definiert werden. Dieses muß am Ende des Dateinamens zusätzlich die Kennung *_P* erhalten. Soll die User-DLL auch auf der Palettenseite *User* der Systemblock-Toolbar erscheinen, muß zusätzlich ein 18x18-Pixel großes Bitmap definiert werden, welches die Kennung *_T* enthält.

Beispiel:	zu MOTOR.DLL	gehört Drucker-Bitmap	MOTOR_P.BMP
		und Toolbar-Bitmap	MOTOR_T.BMP

Entwurf von PID-Reglern

Neben der reinen Simulation von (Regelungs-) Systemen bietet BORIS Ihnen die Möglichkeit zum direkten Entwurf von PID-Reglern. Dabei haben Sie zwei Möglichkeiten:

- Entwurf von PID-Reglern auf der Basis von sogenannten *Einstellregeln*. Diese Entwurfsmethodik ist Bestandteil dieses Kapitels.
- Entwurf von PID-Reglern durch *numerische Parameteroptimierung* mit Hilfe des *BORIS-Parameter-Optimierungsmoduls* (siehe Abschnitt *Numerische Optimierung von Systemen*)

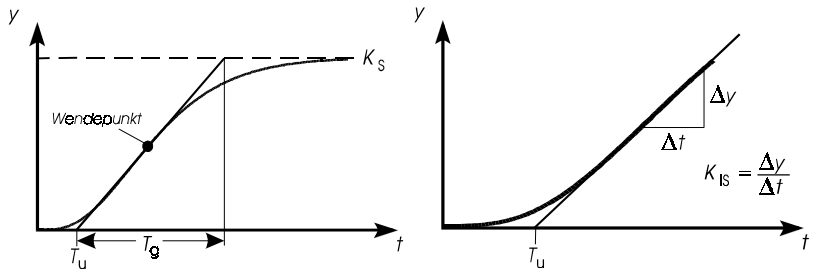
PID-Entwurf nach Einstellregeln

Kenngrößen der Regelstrecke

Die Grundlage des in BORIS realisierten PID-Entwurfs nach Einstellregeln bildet die *Sprungantwort* der Regelstrecke, d. h. der Verlauf der Ausgangsgröße der Strecke bei einem Sprung der Höhe eins an ihrem Eingang. Aus dieser Sprungantwort lassen sich dann *Verzugszeit* T_u und *Ausgleichszeit* T_g der Strecke (bei Strecken mit Ausgleich) bzw. nur die Verzugszeit (bei Strecken ohne Ausgleich) ermitteln. Zusammen mit der *Streckenverstärkung* (*Übertragungsbeiwert*) K_S bzw. K_{IS} bilden diese Parameter dann die Grundlage des Reglerentwurfs (siehe nachfolgendes Bild; näheres dazu z. B. in [7]). In der Regel lassen sich die Einstellregeln nur auf nicht schwingfähige Strecken anwenden.

Hinweis: Einstellregeln für "Nicht-Standard-Strecken" befinden sich zur Zeit in Vorbereitung und werden in einer späteren Version verfügbar sein.

Die Ermittlung der Kenngrößen in BORIS kann manuell oder automatisch erfolgen. Dazu ist zunächst die durch Messung aufgenommene, durch Simulation erzeugte oder aus einer externen Datei gelesene Sprungantwort einem Zeitverlauf-Block zuzuführen.



Kenngrößen der Strecke für Strecken mit Ausgleich (links) bzw. ohne Ausgleich (rechts)

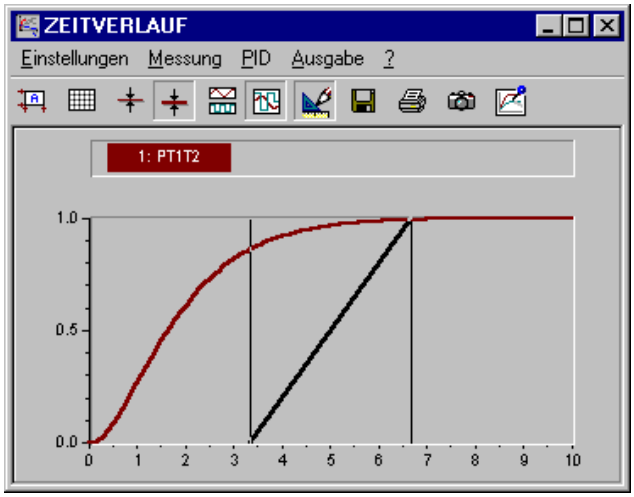
Soll die Kennwertermittlung manuell erfolgen, so muß die Messfunktion des Zeitverlauf-Ausgabefensters aktiviert werden. Außerdem muß über seine Menüoption MESSUNG | TANGENTE EINZEICHNEN die Einzeichnung der Tangente aktiviert werden. Dadurch erscheint im Anzeigefenster zusätzlich zu den Meßcursorn eine Tangente, die ebenfalls mit der Maus verschoben werden kann:

- Zunächst ist die Tangente an die Meßcursor "geheftet" und läßt sich mit diesen verschieben.
- Durch gleichzeitige Betätigung der <Shift>- oder <Strg>-Taste lassen sich die Meßcursor bei Bedarf auch von der Tangente "lösen".

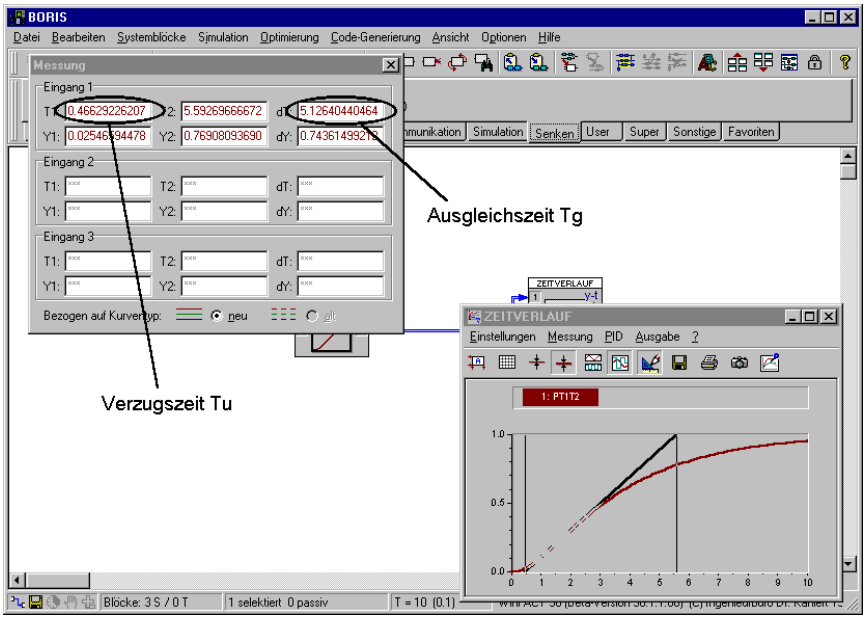
Auf diese Weise läßt sich die Tangente manuell als Wendetangente an die Sprungantwort anlegen. Aus dem Meßwertdialog lassen sich dann die Werte für Verzugszeit bzw. Ausgleichzeit entnehmen (siehe nachfolgendes Bild).

Zur automatischen Kennwertermittlung wird der Menüpunkt PID| PID-ENTWURF... angewählt. Man gelangt dann in den kombinierten Analyse-/Entwurfsdialog. Das obere Gruppenfeld mit dem Titel *Streckenparameter* ist für die Kennwertermittlung maßgebend.

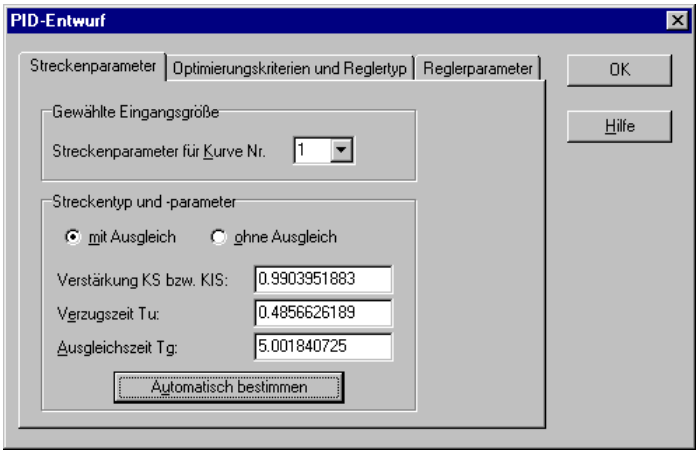
Über das Editierfeld *Streckenparameter für Kurve Nr.* kann bei mehreren dargestellten Kurven zunächst die zu berücksichtigende Sprungantwort ausgewählt werden. Die Kennwertermittlung wird dann über den Schalter *Automatisch bestimmen* gestartet. BORIS ermittelt daraufhin zunächst den Streckentyp (mit/ohne Ausgleich) und im Anschluß daran die Kennwerte der Strecke. Alle Größen können bei Bedarf auch manuell modifiziert werden.



Anzeigefenster des Zeitverlauf-Blocks nach Aktivierung der Meßfunktion und der Tangente



Manuelle Ermittlung von Verzugs- und Ausgleichszeit



Analyse- bzw. Entwurfsdialog (hier nach erfolgter Kennwertermittlung)

Ermittlung der Reglerparameter

Nach erfolgter Kennwertermittlung kann der Reglerentwurf gestartet werden. Dazu wird - falls nicht bereits geschehen - über PID | PID-ENTWURF... in den Analyse-/Entwurfsdialog gewechselt. Zum Entwurf werden die Einstellregeln nach Chien/Hrones und Reswick benutzt (siehe nachfolgende Tabellen). Beim Entwurf ist anzugeben, ob ein Überschwingen der Ausgangsgröße zugelassen werden soll oder nicht und ob besonders gutes Führungs- oder aber Störverhalten erzielt werden soll. Diese Optimierungskriterien und der zu entwerfende Reglertyp sind auf der Palette *Optimierungskriterien und Reglertyp* festzulegen.

Der eigentliche Entwurfsvorgang wird über den Schalter *Reglerparameter berechnen* auf der Palette *Reglerparameter* gestartet. Die ermittelten Reglerparameter werden dann in den entsprechenden Editierfeldern im Gruppenfeld *Reglerparameter* angezeigt und können dort ggf. modifiziert werden. Ist bereits ein PID-Block vorhanden, so können die ermittelten Parameter über den Schalter *Parameter übernehmen* direkt in den ausgewählten PID-Block übernommen werden.

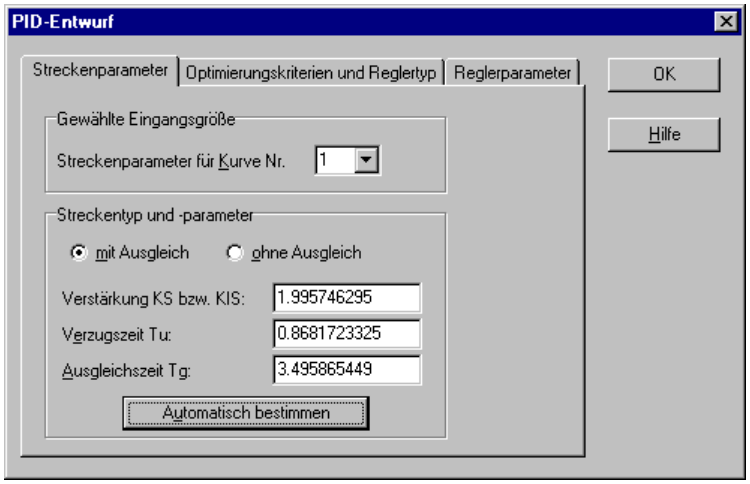
Regler-typ	Mit Überschwingen		Ohne Überschwingen	
	Störung	Führung	Störung	Führung
P	$K_R = 0.71\frac{1}{K_S}\frac{T_g}{T_u}$	$K_R = 0.71\frac{1}{K_S}\frac{T_g}{T_u}$	$K_R = 0.3\frac{1}{K_S}\frac{T_g}{T_u}$	$K_R = 0.3\frac{1}{K_S}\frac{T_g}{T_u}$
PI	$K_R = 0.71\frac{1}{K_S}\frac{T_g}{T_u}$ $T_N = 2.3T_u$	$K_R = 0.59\frac{1}{K_S}\frac{T_g}{T_u}$ $T_N = T_g$	$K_R = 0.59\frac{1}{K_S}\frac{T_g}{T_u}$ $T_N = 4T_u$	$K_R = 0.34\frac{1}{K_S}\frac{T_g}{T_u}$ $T_N = 1.2T_g$
PID	$K_R = 1.2\frac{1}{K_S}\frac{T_g}{T_u}$ $T_N = 2T_u$ $T_V = 0.42T_u$	$K_R = 0.95\frac{1}{K_S}\frac{T_g}{T_u}$ $T_N = 1.35T_g$ $T_V = 0.47T_u$	$K_R = 0.95\frac{1}{K_S}\frac{T_g}{T_u}$ $T_N = 2.4T_u$ $T_V = 0.42T_u$	$K_R = 0.59\frac{1}{K_S}\frac{T_g}{T_u}$ $T_N = 1.35T_g$ $T_V = 0.5T_u$

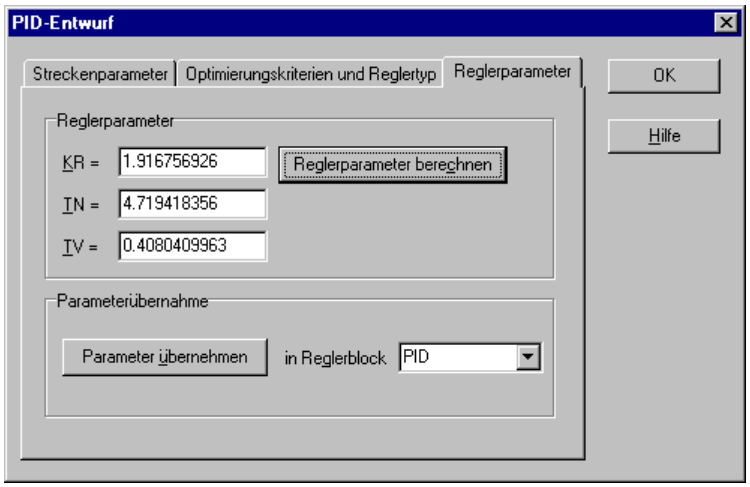
Regler-typ	Mit Überschwingen		Ohne Überschwingen	
	Störung	Führung	Störung	Führung
P	$K_R = 0.71\frac{1}{K_{IS}}\frac{1}{T_u}$	$K_R = 0.71\frac{1}{K_{IS}}\frac{1}{T_u}$	$K_R = 0.3\frac{1}{K_{IS}}\frac{1}{T_u}$	$K_R = 0.3\frac{1}{K_{IS}}\frac{1}{T_u}$
PI	$K_R = 0.71\frac{1}{K_{IS}}\frac{1}{T_u}$ $T_N = 2.3T_u$	$K_R = 0.59\frac{1}{K_{IS}}\frac{1}{T_u}$	$K_R = 0.59\frac{1}{K_{IS}}\frac{1}{T_u}$ $T_N = 4T_u$	$K_R = 0.34\frac{1}{K_{IS}}\frac{1}{T_u}$
PID	$K_R = 1.2\frac{1}{K_{IS}}\frac{1}{T_u}$ $T_N = 2T_u$ $T_V = 0.42T_u$	$K_R = 0.95\frac{1}{K_{IS}}\frac{1}{T_u}$ $T_V = 0.47T_u$	$K_R = 0.95\frac{1}{K_{IS}}\frac{1}{T_u}$ $T_N = 2.4T_u$ $T_V = 0.42T_u$	$K_R = 0.59\frac{1}{K_{IS}}\frac{1}{T_u}$ $T_V = 0.5T_u$

Einstellregeln nach Chien/Hrones und Reswick für Strecken mit (oben) bzw. ohne Ausgleich (unten)

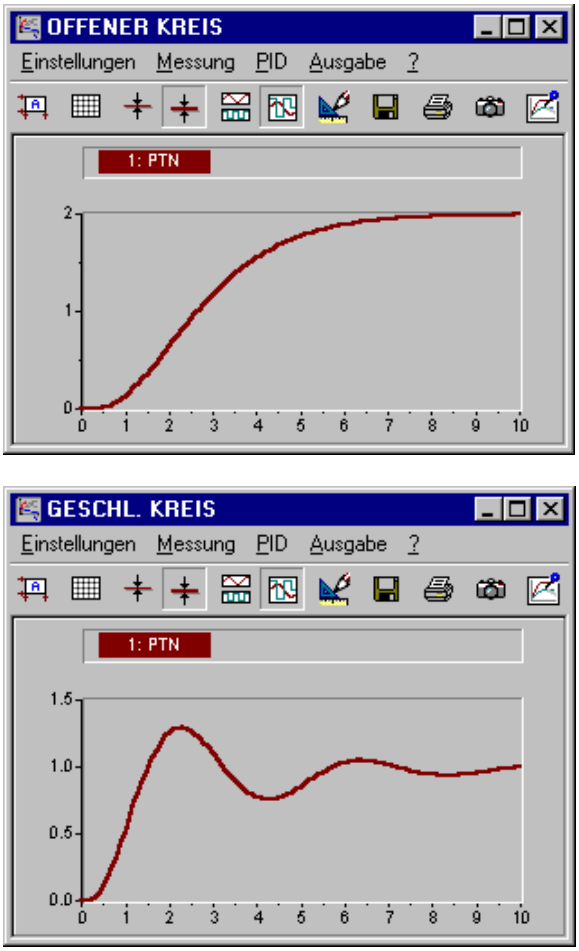
Beispiel

Es soll für eine PT_3-Strecke mit einer dreifachen Zeitkonstanten von 1 und einer Verstärkung von 2 ein PID-Regler entworfen werden, der für gutes Führungsverhalten sorgt. Ein Überschwingen der Regelgröße soll akzeptiert werden. Nachfolgende Grafiken zeigen den Entwurfsdialog nach der Streckenanalyse und dem Reglerentwurf sowie die Sprungantworten der Strecken und des resultierenden geschlossenen Regelkreises.





Streckenanalyse (oben) und Reglerentwurf (unten) für PT_3-Beispielstrecke



Sprungantworten der Regelstrecke (oben) bzw. des geschlossenen Regelkreises (unten)

Hinweis: Das Beispiel befindet sich unter dem Namen PID_DESI.BSY im WinFACT-Beispielverzeichnis.

Numerische Optimierung von Systemparametern

BORIS bietet über die reine Simulation hinaus weitreichende Möglichkeiten zur numerischen Optimierung von Systemparametern mit Hilfe von leistungsfähigen Optimierungsverfahren, sogenannten *Evolutionsstrategien* [14]. Diese sind in der Lage, das gesuchte globale Optimum auch bei sehr zerklüfteten Topologien der Gütefunktion (Zielfunktion) mit einer hohen Zuverlässigkeit zu finden.

Zur Einführung in die numerische Optimierung mit BORIS soll zunächst ein rein mathematisches Optimierungsproblem betrachtet werden. Gesucht seien die Optimierungsparameter $x_1 \ldots x_5$, für die die Gütefunktion

$$Q(x_1,...,x_5) = (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 + (x_4 - 4)^2 + (x_5 - 5)^2$$

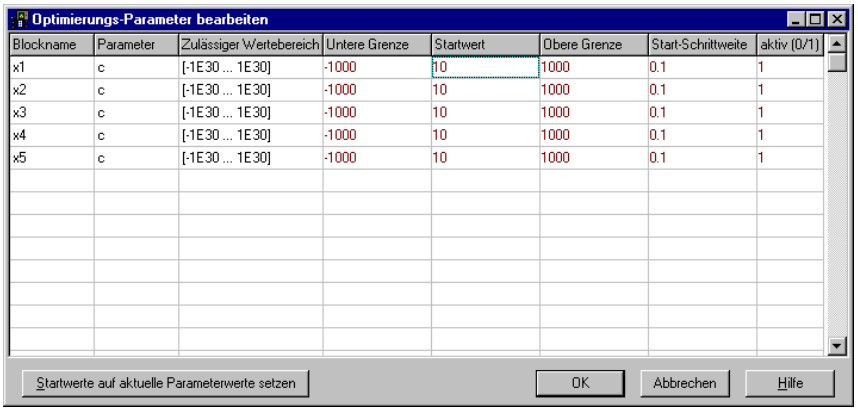
ihr Minimum annimmt. Es sei lediglich bekannt, daß alle Parameter im Bereich $-1000 \le x_i \le 1000$ liegen müssen. Anhand dieses Beispiels läßt sich die Qualität der gefundenen Lösung leicht überprüfen, da das exakte Optimum bekannt ist; es liegt bei

$$x_1 = 1,\ \ x_2 = 2,\ \ x_3 = 3,\ \ x_4 = 4,\ \ x_5 = 5\ .$$

Der zugehörige Wert der Gütefunktion liegt bei null.

Festlegung der Optimierungsparameter

Optimiert werden können alle aktiven Exportparameter der aktuellen Sy-stemebene. Da in obigem Beispiel fünf Parameter zu optimieren sind, wählt man entsprechend fünf Systemblöcke vom Typ *Konstante* und benennt diese mit $x_1 \ldots x_5$. Der Parameter *c* jedes Blocks wird dann als Exportparameter aktiviert. Zur Bearbeitung der Optimierungsparameter dient anschließend die Option OPTIMIERUNG | OPTIMIERUNGSPARAMETER....

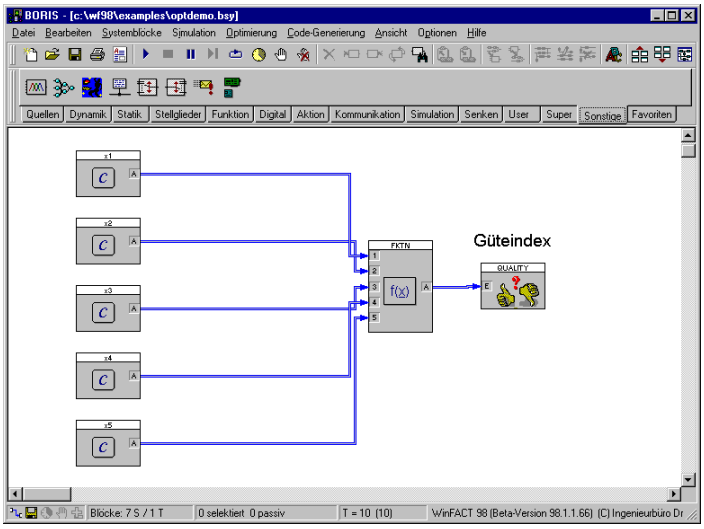


Dialog zur Bearbeitung der Optimierungsparameter

Der zugehörige Dialog erlaubt nun für jeden Optimierungsparameter die Vorgabe der unteren und oberen Grenze, innerhalb der die Optimierung durchgeführt werden soll (in obigem Beispiel -1000 bzw. 1000), des Startwertes für die Optimierung (oben 10) sowie einer Start-Schrittweite (oben 0.1 für alle Parameter). Außerdem können einzelne Parameter von der Optimierung ausgeschlossen werden, indem sie durch einen Eintrag 0 in der Spalte *aktiv (0/1)* deaktiviert werden. Dies ist z. B. sinnvoll, wenn bei einem Block mit drei Exportparametern nur zwei dieser Parameter auch optimiert werden sollen. Über die Schaltfläche *Startwerte auf aktuelle Parameter setzen* ist es möglich, im Anschluß an eine bereits durchgeführte Optimierung einen neuen Lauf zu starten, der mit den Ergebnissen der vorangegangenen Optimierung als Startwerten beginnt.

Definition des Gütekriteriums

BORIS versteht unter einer Optimierung grundsätzlich die *Minimierung* einer Güte- oder Zielfunktion, die innerhalb der zugrundeliegenden Systemstruktur einfach grafisch definiert werden kann. Dazu dient ein sogenannter *Güteindex-Block*, der über OPTIMIERUNG | GÜTEINDEX-BLOCK EINFÜGEN wie ein gewöhnlicher Systemblock eingefügt werden kann. Das in diesen Block einfließende Signal stellt anschließend die Gütefunktion dar. Untenstehende Grafik zeigt die Optimierungsstruktur für das Beispiel-Problem, das als OPTDEMO.BSY im Examples-Verzeichnis zu finden ist. Der Funktionsblock dient dabei zur Berechnung der eigentlichen Gütefunktion $Q(x_1,...,x_5)$.

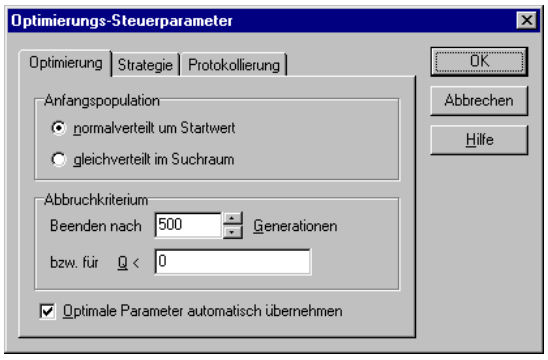


Optimierungsstruktur für Beispiel-Problem

Steuerparameter für die Optimierung

Das Optimierungsverfahren auf der Basis von Evolutionsstrategien (Genetischen Algorithmen) besitzt eine Reihe von Steuerparametern (Strategieparametern), über die die Optimierung beeinflußt werden kann. Diese sind über OPTIMIERUNG | STEUERPARAMETER... zugänglich. Der zugehörige Dialog ist in drei Palettenseiten aufgeteilt.

Palettenseite *Optimierung*



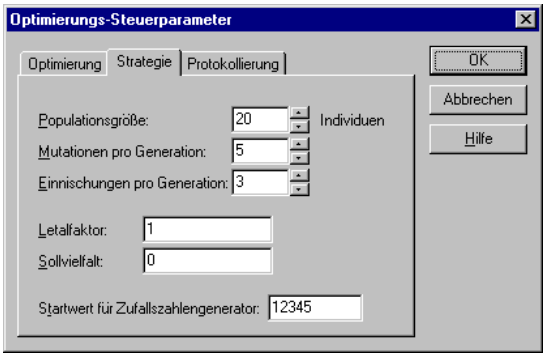
Diese Palettenseite umfaßt folgende Optionen:

Anfangspopulation — Diese Einstellung legt die Art der Anfangspopulation fest. In der Einstellung *normalverteilt um Startwert* werden alle Individuen normalverteilt um den unter *Optimierungsparametern* eingestellten Startwert generiert. Die Streuung der Normalverteilung entspricht dabei der gewählten Anfangsschrittweite. In der Einstellung *gleichverteilt im Suchraum* wird die Anfangspopulation gleichtverteilt innerhalb der vorgegebenen Parametergrenzen erzeugt.

Abbruchkriterium — Legt das Abbruchkriterium für die Optimierung fest. Diese wird beendet, wenn eine vorgebbare Zahl von Generationen überschritten wurde oder die Gütefunktion einen bestimmten Grenzwert unterschreitet.

Optimale Parameter automatisch übernehmen — Ist diese Option aktiviert, werden nach Beendigung der Optimierung die ermittelten optimalen Parameter automatisch in die entsprechenden Blockparameter übernommen.

Palettenseite *Strategie*

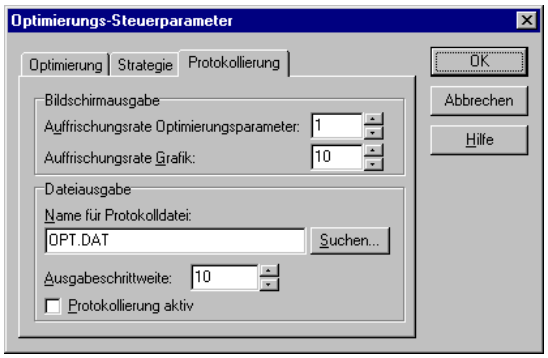


Diese Palettenseite umfaßt folgende Optionen:

Populationsgröße — Anzahl der Individuen in der Population. Dieser Wert sollte an die Anzahl der Optimierungsparameter angepaßt werden und mindestens vier bis fünf mal so groß sein.

Mutationen pro Generation	Legt die Anzahl der mutierten Individuen pro Generation fest. Ein höherer Wert führt zu einer mehr globalen Suche nach dem Optimum, kann aber zu einer verlangsamten Konvergenz führen (siehe dazu [14]).
Einnischungen pro Generation	Legt die Anzahl der Einnischungen pro Generation fest. Ein höherer Wert führt zu einer mehr globalen Suche nach dem Optimum, kann aber zu einer verlangsamten Konvergenz führen (siehe dazu [14]).
Letalfaktor	Legt den Letalfaktor für die Mutationen fest. Ein niedriger Wert führt zu einer mehr globalen Suche nach dem Optimum, kann aber zu einer verlangsamten Konvergenz führen (siehe dazu [14]).
Sollvielfalt	Legt die Sollvielfalt der Individuen innerhalb der Population fest. Ein höherer Wert führt zu einer mehr globalen Suche nach dem Optimum, kann aber zu einer verlangsamten Konvergenz führen (siehe dazu [14]).
Startwert für Zufallszahlengenerator	Legt den Startwert für den Zufallszahlengenerator fest, der u. a. zur Erzeugung von Mutationen benutzt wird.

Palettenseite *Protokollierung*



Diese Palettenseite umfaßt folgende Optionen:

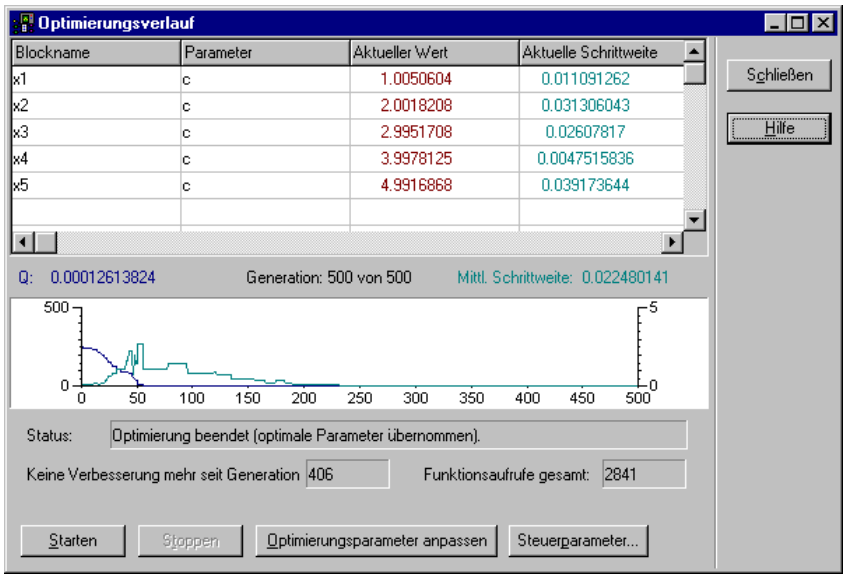
Bildschirmausgabe	Legt die Refreshraten für die Bildschirmausgabe fest

(siehe Abschnitt *Steuern des Optimierungsablaufs*)

Dateiausgabe Diese Einstellungen ermöglichen die Protokollierung des Optimierungsverlaufs in einer Textdatei. Der im Feld *Ausgabeschrittweite* angegebene Wert legt bei einer aktiven Protokolldatei fest, nach jeder wievielten Generation ein Abspeichern erfolgt.

Steuern des Optimierungsablaufs

Sind alle Vorbereitungen beendet, gelangt man über OPTIMIERUNG | OPTIMIERUNG STARTEN... in den Dialog zur Optimierungssteuerung. Nachstehende Bildschirmgrafik zeigt den Dialog nach einem erfolgreichen Optimierungslauf für das behandelte Beispiel-Problem.



Dialog zur Optimierungssteuerung (hier nach einer bereits beendeten Optimierung)

Dieser Dialog zeigt im oberen Teil alle Optimierungsparameter mit ihren aktuellen Werten (rot) und Schrittweiten (blaugrün) an. Die Grafik im mittleren Teil zeigt während der Optimierung den zeitlichen Verlauf der Gütefunktion und der mittleren Schrittweite an. Zur weiteren Information dient die Anzeige der letzten erfolgreichen Generation und der Gesamtzahl der Funktionsaufrufe, d. h. Simulationsläufe.

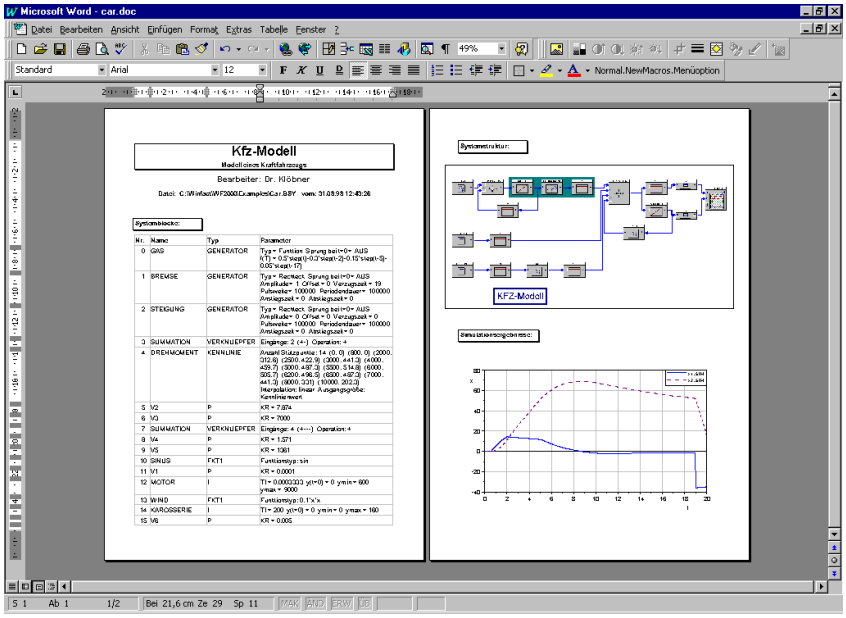
Ein neuer Optimierungslauf wird über die Schaltfläche *Starten* gestartet und kann dann jederzeit über die *Stoppen*-Schaltfläche abgebrochen werden. Nach Beendigung des Laufs können über *Optimierungsparameter anpassen* die aktuellen Optimierungsparameter als neue Startwerte und die aktuellen Schrittweiten als neue Start-Schrittweiten übernommen werden, um basierend auf diesen Einstellungen einen neuen Optimierungslauf zu starten.

Anwendung zur Regleroptimierung

Eine wesentliche Anwendung der numerischen Parameteroptimierung liegt in der automatischen Optimierung von Reglerparametern, beispielsweise nach dem ITAE-Kriterium oder beliebigen Gütekriterien. Dabei kann auf einfache Weise z. B. auch der maximale Stellgrößenbedarf des Reglers im Gütekriterium berücksichtigt werden. Ebenso können mehrere Regler simultan, d. h. parallel optimiert werden. Zwei Anwendungsbeispiele hierzu finden Sie im Examples-Verzeichnis in den Dateien ITAEOPT.BSY und ITAEOPT2.BSY.

Dokumentation von Systemen

BORIS ermöglicht über seinen integrierten Dokumentgenerator die komfortable Dokumentation einer kompletten Systemstruktur sowohl in grafischer Form als auch in Textform. Dabei kann die Ausgabe sowohl direkt auf dem Drucker als auch in einer Vielzahl von Ausgabeformaten erfolgen, so daß eine Weiterverarbeitung mit anderen Standardanwendungen möglich ist.

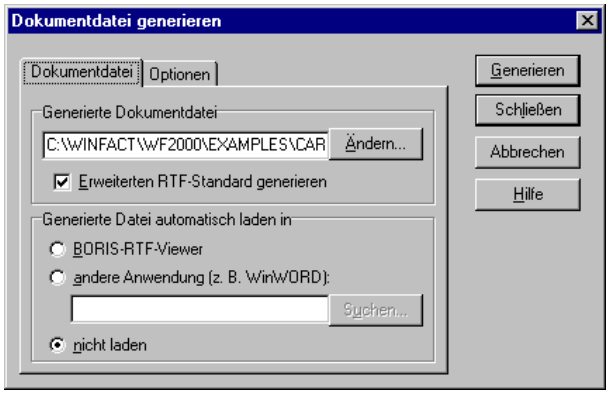


Dokumentation eines Systems mit dem integrierten BORIS-Dokumentgenerator

Erzeugung der Dokumentdatei

Sobald eine BORIS-Systemdatei bzw. Superblockdatei einmal gespeichert wurde, kann für diese Datei eine Dokumentdatei generiert werden. Dies ist eine Textdatei im RTF-Format (*Rich Text Format*), die von praktisch jedem Textverarbeitsprogramm gelesen und weiterverarbeitet werden kann. Steht ein solches Textverarbeitungprogramm nicht zur Verfügung, kann auch der integrierte RTF-Viewer benutzt werden. Dieser ist allerdings von seinem Leistungsumfang her gegenüber Standard-Textverarbeitungsprogrammen stark eingeschränkt.

Zur Generierung der Dokumentation wählen Sie DATEI | DOKUMENTDATEI GENERIEREN... bzw. die Schaltfläche  der System-Toolbar. Es folgt ein in mehrere Paletten aufgeteilter Dialog, über den die Dokumentdatei-Generierung gesteuert werden kann.



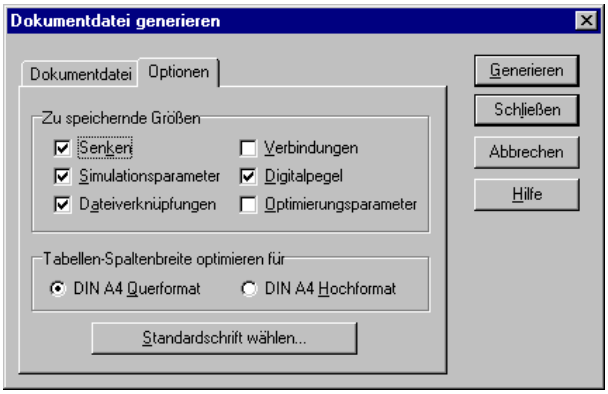
Dialog zur Steuerung der Dokumentgenerierung (Palette Dokumentdatei*)*

Im Eingabefeld *Generierte Dokumentdatei* kann der Name der generierten RTF-Datei angegeben werden. Dieser entspricht per Voreinstellung dem Namen der aktuellen Systemdatei, jedoch mit der Extension RTF. Ist das Optionsfeld *Erweiterten RTF-Standard generieren* ausgewählt, so wird die Datei in Tabellenform dargestellt, was zu einer wesentlich übersichtlicheren Darstellung führt. Der integrierte RTF-Viewer sowie einige einfache Texteditoren beherrschen diesen erweiterten Standard jedoch nicht, so daß die Option in diesen Fällen deaktiviert werden sollte.

Auf Wunsch kann die generierte Datei direkt in den entsprechenden RTF-Editor bzw. das Textverarbeitungsprogramm geladen werden. Dazu ist die zugehörige Option im Gruppenfenster *Generierte Datei automatisch laden in* zu aktivieren.

Über die Palette *Optionen* des Dialogs können zunächst die in der Dokumentdatei abzuspeichernden Größen ausgewählt werden (Gruppenfenster *Zu speichernde Größen*). Im Gruppenfenster *Tabellen-Spaltenbreite optimieren für* wird das zugrundeliegende Papierformat für die Datei festgelegt. Über die Schaltfläche *Standardschrift wählen* kann schließlich der Schrifttyp für die Ausgabe festgelegt werden.

Die eigentliche Dokumentgenerierung wird über die Schaltfläche *Generieren* eingeleitet. Im Anschluß daran wird – sofern diese Option nicht zuvor deaktiviert wurde – die generierte Datei automatisch in die gewählte Textverarbeitung geladen.



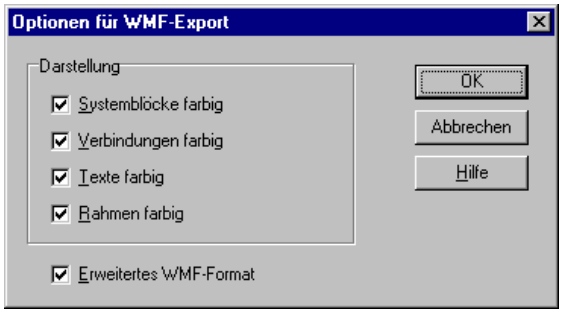
Palette Optionen

Exportieren der Systemstruktur

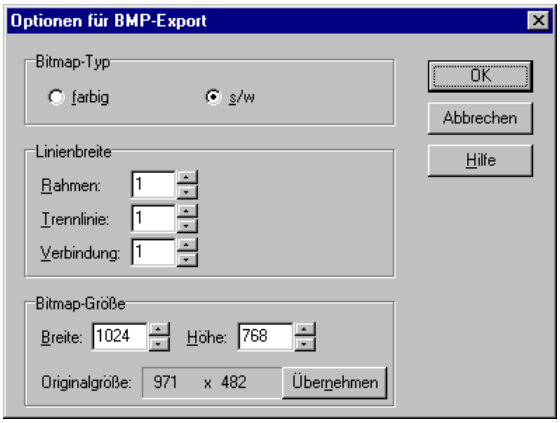
Die Systemstruktur kann wahlweise in zwei Formaten exportiert werden:

- Als Bitmap-Grafik im BMP-Format
- Als Vektorgrafik im WMF-Format

Beide Formate können sowohl farbig als auch im s/w-Format exportiert werden und sind praktisch von allen Windows-Anwendungen (Textverarbeitung, Grafikprogramm, Präsentationsprogramm usw.) weiterverarbeitbar. Zum Export dient die Menüoption DATEI | EXPORTIEREN.... Im Anschluß daran erscheint ein Windows-Dateiauswahldialog, über den das gewünschte Exportformat und der Dateiname festgelegt werden können. Abhängig vom gewählten Format kann nachfolgend die Exportgrafik genauer spezifiziert werden.



Export-Optionen für WMF-Format

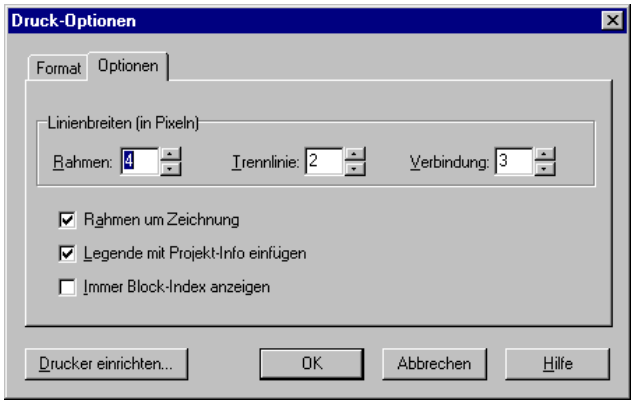


Export-Optionen für BMP-Format

Drucken der Systemstruktur

Alternativ zum Export der Systemstruktur kann diese auch direkt aus BORIS heraus gedruckt werden. Dazu dient die Menüoption DATEI | DRUCKEN... bzw. die Schaltfläche der System-Toolbar. Der Ausdruck kann durch eine Reihe von Optionen gesteuert werden, die über einen entsprechenden Dialog vorgegeben werden können.

Druck-Optionen-Dialog, Palette Format



Druck-Optionen-Dialog, Palette Optionen

Die einzelnen Optionen haben folgende Bedeutungen:

Option	**Bedeutung**
Druckmodus	Legt die Größe des Ausdrucks sowie die Farbauflösung fest
Druckformat	Legt fest, ob der Ausdruck mit einer festen Größe - ggfls. verteilt auf mehrere Seiten - erfolgen oder aber automatisch an die Seitengröße angepaßt werden soll. Bei einer mehrseitigen Ausgabe können die Seitenumbrüche durch Aktivierung der Option *Seitenumbrüche auf Arbeitsblatt anzeigen* bereits zur Entwurfszeit kontrolliert werden. Die aktuelle Seitenaufteilung wird ferner im Feld *Seitenaufteilung* angezeigt.
Linienbreiten in Pixeln	Ermöglicht die Vorgabe der Linienbreiten für die verschiedenen Komponenten der Systemstruktur
Rahmen um Zeichnung	Legt fest, ob ein Rahmen um die Grafik gedruckt werden soll
Legende mit Projekt-Info einfügen	Ist diese Option aktiviert, enthält die Grafik automatisch eine Legende mit den wichtigsten Daten der zugehörigen Projekt-Information.
Immer Block-Index anzeigen	Ist diese Option aktiviert, wird unabhängig von der Einstellung für die Bildschirmausgabe beim Drucken grundsätzlich der Block-Index angezeigt.

11 Das grafische Präsentationsmodul INGO

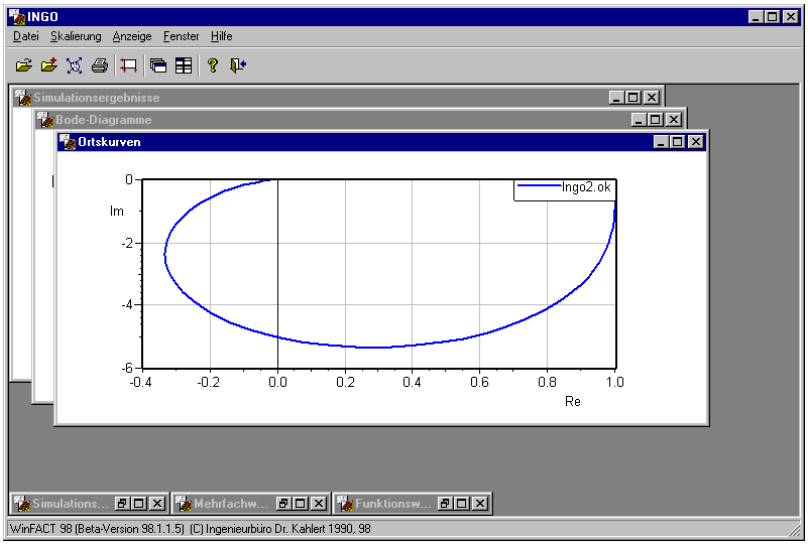
Leistungsumfang

Das WinFACT-Modul INGO ermöglicht die grafische Darstellung aller WinFACT-typischen Grafikdateien:

- Simulationsergebnisse (SIM-Dateien)
- Allgemeine Wertepaare (XY-Dateien)
- Mehrfachwertepaare (MXY-Dateien)
- Bode-Diagramme (BD-Dateien)
- Nyquist-Ortskurven (OK-Dateien)
- Funktionswertmatrizen (FWM-Dateien, Darstellung wahlweise in Höhenlinien- oder 3D-Kennfeldform)

Zur Weiterverarbeitung bietet INGO eine Exportfunktion, die die jeweilige Grafik als WMF-Datei (Windows Meta File) abspeichert. Somit ist eine Weiterverarbeitung ohne Qualitätsverlust möglich. Die Druckerausgabe von INGO kann derart erfolgen, daß Achsenbeschriftungen sich mit Millimeterangaben auf dem zu bedruckenden Blatt decken. Dadurch können in dem Ausdruck mit Hilfe eines Lineals leicht weitere Messungen vollzogen werden.

Die Benutzerschnittstelle basiert auf dem Windows-Multi-Document-Interface und erlaubt damit die gleichzeitige Darstellung mehrerer Grafikfenster auch unterschiedlichen Typs, wobei jedes Grafikfenster wiederum mehrere Kurven enthalten kann. Jede Kurve kann wahlweise in Linienform oder in Form von Markierungssymbolen dargestellt werden. Jedes Grafikfenster kann - dem MDI-Standard folgend - beliebig verschoben, in der Größe verändert oder zum Symbol verkleinert werden. Das nachfolgende Bild zeigt ein typisches Hauptfenster des Programms mit mehreren Dokumentfenstern, die teilweise zum Symbol verkleinert wurden.



Hauptfenster von INGO mit mehreren unterschiedlichen Grafikdokumenten

Programmoptionen

Öffnen eines neuen Grafikfensters

Ein neues Grafikfenster wird über die Menüfolge DATEI | ÖFFNEN, die Tastenkombination <Strg><N> oder die Toolbar geöffnet. Gleichzeitig muß der Name der ersten darzustellenden Datei angegeben werden. Der Typ dieser Datei (z. B. SIM) legt automatisch den Typ des Grafikfensters fest, so daß beim späteren Hinzufügen neuer Kurven jeweils nur die passenden Dateien angeboten werden.

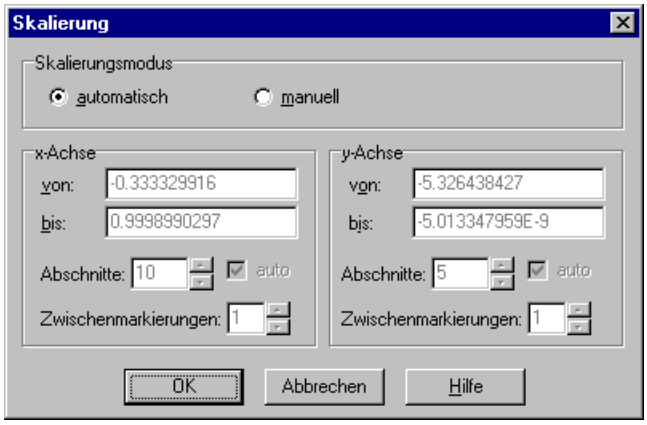
Hinzufügen weiterer Kurven

In das jeweils aktive Grafikfenster können durch DATEI | HINZUFÜGEN oder die entsprechende Schaltfläche in der Werkzeugleiste weitere Kurven vom gleichen Typ hinzugefügt werden. Ein Grafikfenster vom Typ SIM, XY, MXY, BD oder OK kann beliebig viele Kurven enthalten. FWM-Grafiken in Höhenliniendarstellung können ebenfalls beliebig viele Dateien aufnehmen, wobei jede Teilgrafik maximal 100 Höhenlinienwerte umfassen darf. Lediglich bei 3D-Kennfeldern auf der Basis von FWM-Dateien ist jeweils nur ein Kennfeld pro Grafikfenster möglich.

Skalierung der Grafikachsen

Die Skalierung der Achsen wird automatisch in der Weise aktualisiert, daß alle Kurven vollständig dargestellt werden. Eine Modifikation der Achsenskalierung ist über den Menüpunkt SKALIERUNG | SKALIERUNG und den entsprechenden Eingabedialog möglich.

Hinweis: FWM-Dateien zur Darstellung von Funktionen zweier Veränderlicher enthalten keine Informationen über den *x*- bzw. *y*-Wertebereich (siehe Kapitel *Grundlagen*, Abschnitt *WinFACT-Dateiformate*)! Daher werden *x*- und *y*-Achsen gemäß der Anzahl der enthaltenen Werte skaliert, bei einer 10×10-Funktionswertematrix z. B. also jeweils von 0 bis 10. Eine Umskalierung der Achsen hat bei diesem Grafiktyp also keinerlei Auswirkung auf das Kennfeld bzw. die Höhenlinien selbst!



Dialog zum Skalieren der Achsen für Grafiken vom Typ XY, MXY, SIM, FWM oder OK. Bei manueller Skalierung kann die Anzahl der Achsenabschnitte automatisch oder manuell (dann mit freier Wahl der Zwischenmarkierungen) gewählt werden.

Dialog zur Skalierung von Grafiken des Typs BD

Anzeige

Beschriftung von Grafiken

Über ANZEIGE | BESCHRIFTUNGEN kann eine individuelle Beschriftung einzelner Grafikfenster vorgenommen werden.

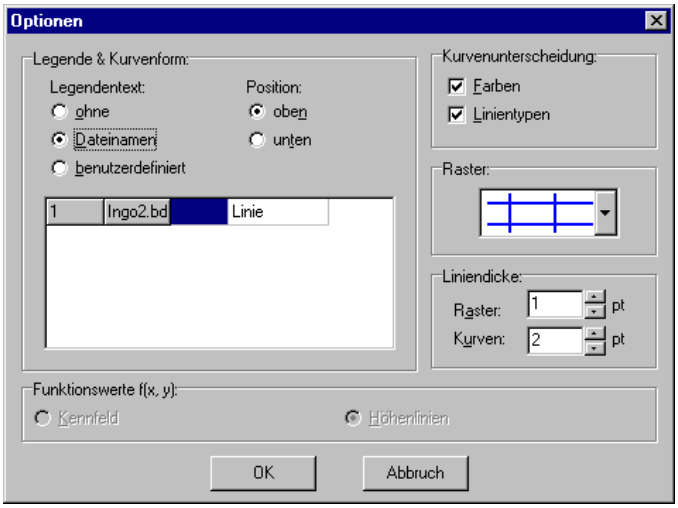
Dialog zur Beschriftung von Grafiken

Folgende Optionen stehen zur Verfügung:

- *Fenstertitel* setzt den Titelbalken des Fensters auf den vorgegebenen Text
- *Titel* plaziert einen vorgebbaren Text innerhalb der Grafik am oberen Bildrand
- *Ordinate links* setzt die Beschriftung der linken Ordinate
- *Ordinate rechts* setzt die Beschriftung der rechten Ordinate bei Bode-Diagrammen
- *Abszisse* setzt die Beschriftung der Abszisse auf den vorgegebenen Text

Anzeigeoptionen

Über ANZEIGE | OPTIONEN wird ein Dialog verfügbar, der die Einstellung weiterer Grafikfenster-spezifischer Optionen erlaubt.



Dialog zum Einstellen von diversen Optionen

Der Dialog enthält folgende Wahlmöglichkeiten:

- Wahl von Legende und Kurventypen

  Die Kurven können wahlweise ohne Legende, mit ihrem Dateinamen oder mit einer benutzerdefinierten Legende dargestellt werden. Außerdem kann für jede Datei festgelegt werden, ob die Werte in Form einer Kurve oder als einzelne Markierungssymbole (z. B. für die Darstellung von Meßwerten) eingezeichnet werden sollen (Ausnahme: FWM-Dateien).

- Kurvenunterscheidung

  Diese beiden Schalter geben an, ob unterschiedliche Kurven eines Diagramms in unterschiedlichen Farben und/oder unterschiedlichen Linientypen dargestellt werden sollen. Die Kurvenfarben können im WinFACT-Setup-Programm WFSETUP eingestellt werden (siehe Kapitel *Erste Schritte*).

- Raster

  Über diese Listbox wird die Art des Koordinatenrasters festgelegt.

- Liniendicke

  Hier läßt sich die Liniendicke für Kurven und Raster festlegen. Erlaubt sind Liniendicken zwischen 1 und 5 Pixeln.

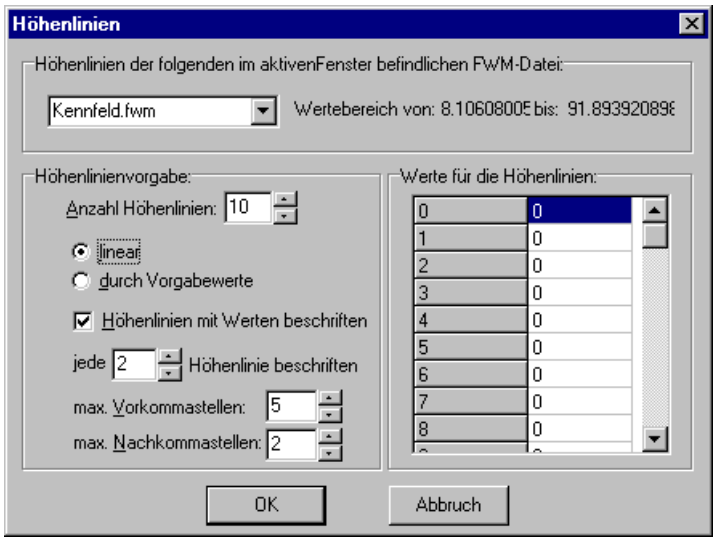
- Funktionswerte f(x, y)

Legt fest, ob eine FWM-Datei in Form von Höhenlinien *f = const* oder als 3D-Kennfeld dargestellt werden soll.

Spezielle Optionen für Höhenlinien bzw. Kennfelder

Für die Darstellung von FWM-Dateien in Höhenlinien- bzw. Kennfeldform stehen einige zusätzliche Optionen zur Verfügung, die über OPTIONEN | HÖHENLINIEN... bzw. OPTIONEN | KENNFELD... erreichbar sind.

Für Höhenliniendarstellung gelangen Sie in folgenden Dialog:



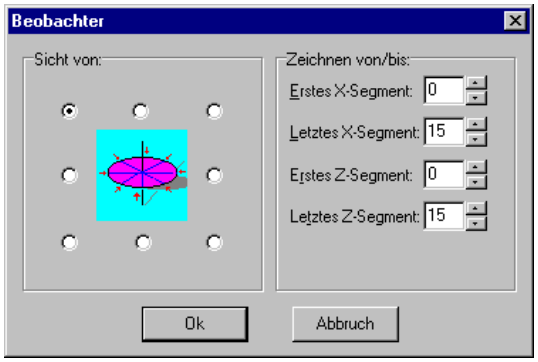
Spezielle Optionen für Höhenlinien

Der Dialog legt die Funktionswerte fest, für die Höhenlinien ermittelt werden. Das zu modifizierende Höhenlinienfeld wird dazu zunächst über das Auswahlfeld in der oberen linke Ecke des Dialogs ausgewählt:

- In der Einstellung *linear* wird die über *Anzahl Höhenlinien* angegebene Zahl von Höhenlinien gezeichnet, wobei die Höhenlinienwerte linear im Wertebereich der Funktion aufgeteilt werden. Der Wertebereich wird hinter dem Auswahlfeld angezeigt.
- In der Einstellung *durch Vorgabewerte* können die gewünschten Werte im rechten Gruppenfeld in die entsprechenden Eingabefelder eingetragen werden.
- Optional können die Höhenlinien mit Werten beschriftet werden.

- Häufig ist es ratsam, gerade bei der Anzeige von sehr vielen Höhenlinien, nicht jede zu beschriften. Daher kann angegeben werden, jede wievielte Höhenlinie in welcher Form (Vor- und Nachkommastellenanzahl) beschriftet werden soll.

Im Falle der Kennfelddarstellung erscheint folgender Dialog:



Optionen für Kennfelder

Dieser Dialog erlaubt eine Änderung des Betrachterstandorts für das Kennfeld sowie ein Ausblenden einzelner Segmente (Netzstücke).

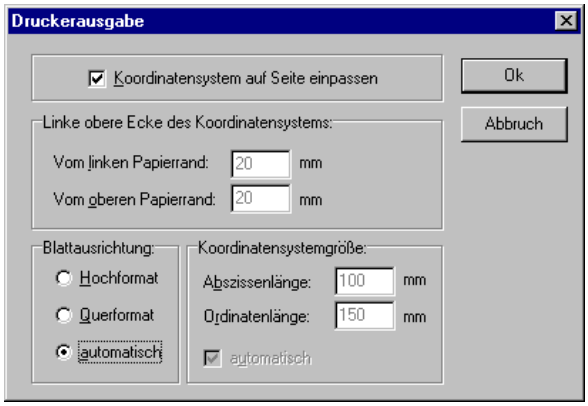
Ausgabefunktionen

Exportfunktion

Die Exportfunktion von Ingo, DATEI | EXPORT, basiert auf einem unter Windows als Standard-Grafik-Dateiformat anzusehendem Format. Die Dateiendung für dieses Format wurde auf WMF (Windows Meta File) festgelegt. Es kann mit allen gängigen Grafikprogrammen gelesen werden. Das WMF-Dateiformat ist ein vektororientiertes Format, das heißt, es wird die Information zum Aufbau des Bildes gespeichert und nicht das Bild selbst. Sie können eine gezielte Veränderung einzelner Bildelemente vornehmen, ohne daß ein Qualitätsverlust einhergeht.

Druckerausgabe

Mit Hilfe der Einstellungen zum Ausdrucken, DATEI | DRUCKEN, eines Anzeigefensters kann ein maßstabsgetreuer Ausdruck erzielt werden (nicht bei der Anzeige von Kennfeldern in der 3D-Ansicht).



Einstellungen für die Ausgabe auf dem Drucker

Der Dialog hat zwei grundlegende Einstellungen:

- Die Einstellung *Koordinatensystem auf Seite einpassen* richtet sich nach der aktuell ausgewählten Blattgröße des Druckers. Das auszudruckende Koordinatensystem wird so gedruckt, daß es möglichst den Platz der gesamten Blattgröße einnimmt. Die Ausrichtung des Blattes kann vorgegeben (*Hochformat* oder *Querformat*) oder *automatisch* von Ingo bestimmt werden.
- Die andere Möglichkeit besteht darin, daß Sie Position und Größe des Koordinatensystems vorgeben. Die vorzugebende Position bezieht sich auf den linken oberen Papierrand, die Größe auf die Achsenlängen in Millimetern. Durch die geeignete Wahl der Größe kann ein ganzzahliges Teilungverhältnis zur Achsenskalierung im Anzeigefenster erzielt werden, was zu einer maßstabsgetreuen Abbildung beim Ausdruck führt. Die Achsenlängen können auch automatisch von INGO bestimmt werden. Dabei wird versucht den Rest des Blattes unter Berücksichtigung eines vernünftigen Maßstabes, so gut wie möglich auszufüllen.

12 Beispiel- und Übungskatalog

Sinn und Zweck dieses Beispielkatalogs

Der folgende Aufgabenkatalog enthält eine Vielzahl unterschiedlicher Aufgaben, die der Übersichtlichkeit halber in folgende Kategorien aufgeteilt sind:

I. Lineare Systeme

II. Reglerentwurf

III. Simulation

IV. Fuzzy-Logik

V. Fuzzy-Control

VI. Meßtechnik

VII. Digitaltechnik

VIII. Systemidentifikation

Alle Aufgaben lassen sich mit Hilfe des Programmsystems WinFACT, evtl. in Verbindung mit etwas "Handarbeit", lösen. Bei der Zusammenstellung der Übungsbeispiele wurde versucht, möglichst alle Themenbereiche anzuschneiden, so daß jeder Anwender "sein" Gebiet in den Beispielen wiederfindet.

Sinn dieses Katalogs ist es, einerseits einen Einblick in den Leistungsumfang von WinFACT zu geben, andererseits aber auch Hilfestellung bei der Einarbeitung in das Programmsystem zu bieten. Derjenige, der WinFACT in der Lehre einsetzen möchte, findet hier ferner einen Grundstock an Übungsaufgaben sowie Anregungen für die Formulierung eigener Problemstellungen[*].

Alle Aufgaben bestehen aus der Aufgabenformulierung, einer Lösungsskizze, die alle notwendigen Hinweise zur Ermittlung der Lösung enthält, sowie bei umfangreicheren Problemstellungen zusätzlich aus den zur Ermittlung der Lösung erforderlichen Systemdateien, die sich auf der Beispieldiskette befinden.

Hinweis: Die Abbildungen enthalten größtenteils noch Bildschirmgrafiken aus älteren WinFACT - Versionen. Dies spielt aber für das Lösungsprinzip keine Rolle.

[*] die wir dann gerne zur Erweiterung dieses Katalogs entgegennehmen!

Kategorie I: Lineare Systeme

Aufgabe I.1: Sprungantwort eines PT1-Gliedes

Aufgaben-stellung:

Gegeben sei ein PT_1-Glied mit der Eingangsgröße $u(t)$ und der Ausgangsgröße $y(t)$. Das System werde beschrieben durch die Übertragungsfunktion

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K}{1+Ts}$$

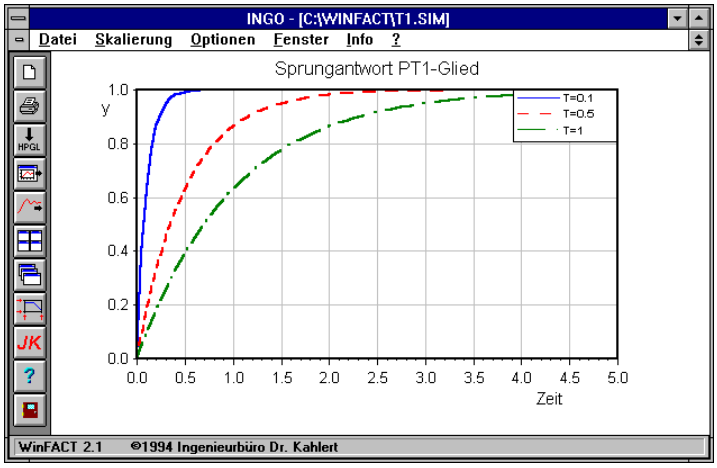
bzw. die Differentialgleichung

$$T\dot{y} + y = Ku$$

Ermitteln Sie die Sprungantwort des Systems für den Fall $K = 1$ und die Zeitkonstanten T = 0.1, 0.5 bzw. 1. Wählen Sie dazu eine Simulationsdauer von 5.

Lösungs-skizze:

Die Aufgabe verdeutlicht den Einfluß der Zeitkonstanten auf das dynamische Verhalten des Systems. Die Lösung kann mit Hilfe von LISA ermittelt werden. Die ermittelten Kurven können abgespeichert und dann mit INGO verglichen werden (s. nachfolgende Grafik).



Aufgabe I.2: Frequenzgang eines PT1-Gliedes

Aufgaben-stellung: Gegeben sei ein PT_1-Glied mit der Eingangsgröße $u(t)$ und der Ausgangsgröße $y(t)$. Das System werde beschrieben durch die Übertragungsfunktion

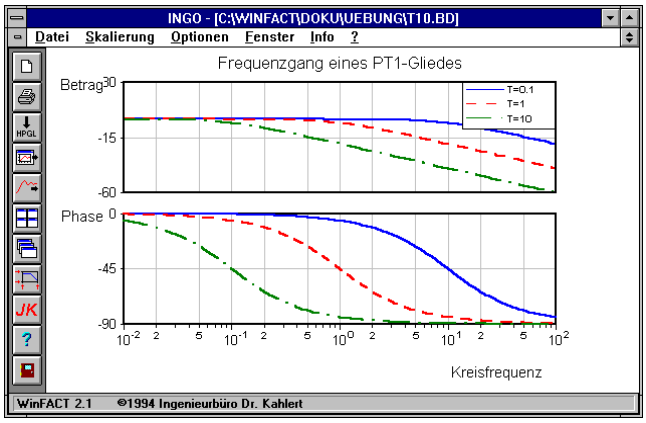
$$G(s) = \frac{Y(s)}{U(s)} = \frac{K}{1+Ts}$$

bzw. die Differentialgleichung

$$T\dot{y} + y = Ku$$

Ermitteln Sie den Frequenzgang des Systems für den Fall K=1 und die Zeitkonstanten T = 0.1, 1 bzw. 10 im Bereich $0.01 \le \omega \le 100$.

Lösungs-skizze: Die Aufgabe verdeutlicht den Einfluß der Zeitkonstanten, d. h. der Eckfrequenz, auf den Frequenzgang des Systems. Die Lösung kann mit Hilfe von LISA ermittelt werden. Die ermittelten Kurven können abgespeichert und dann mit INGO verglichen werden (s. nachfolgende Grafik).



Aufgabe I.3: Sprungantwort eines PT_2-Gliedes

Aufgaben-stellung: Gegeben sei ein PT_2-Glied mit der Eingangsgröße $u(t)$ und der Ausgangsgröße $y(t)$. Das System werde beschrieben durch die Übertragungsfunktion

$$G(s) = \frac{K}{\left(\frac{s}{\omega_n}\right)^2 + 2\frac{\zeta}{\omega_n}s + 1}$$

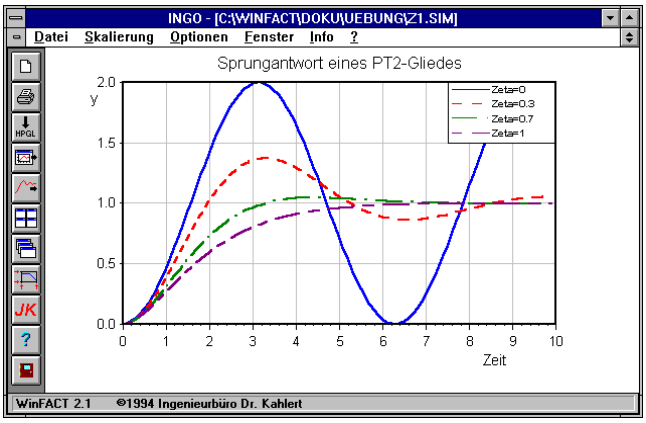
bzw. die Differentialgleichung

$$\frac{1}{\omega_n^2}\ddot{y} + 2\frac{\zeta}{\omega_n}\dot{y} + y = Ku$$

Ermitteln Sie die Sprungantwort des Systems für den Fall $K = \omega_n = 1$ und Dämpfungswerte von $\zeta = 0,\ 0.3,\ 0.7,\ 1$. Wählen Sie dazu eine Simulationsdauer von 10.

Lösungsskizze:

Die Aufgabe verdeutlicht den Einfluß der Dämpfung ζ auf das dynamische Verhalten des Systems. Die Lösung kann mit Hilfe von LISA ermittelt werden. Die ermittelten Kurven können abgespeichert und dann mit INGO verglichen werden (s. nachfolgende Grafik).



Aufgabe I.4: Frequenzgang eines PT_2-Gliedes

Aufgabenstellung:

Gegeben sei ein PT_2-Glied mit der Eingangsgröße $u(t)$ und der Ausgangsgröße $y(t)$. Das System werde beschrieben durch die Übertragungsfunktion

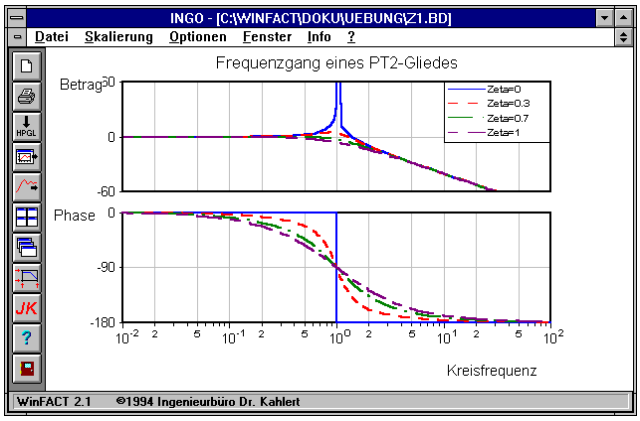
$$G(s) = \frac{K}{\left(\frac{s}{\omega_n}\right)^2 + 2\frac{\zeta}{\omega_n}s + 1}$$

bzw. die Differentialgleichung

$$\frac{1}{\omega_n^2}\ddot{y} + 2\frac{\zeta}{\omega_n}\dot{y} + y = Ku$$

Ermitteln Sie den Frequenzgang des Systems für den Fall $K = \omega_n = 1$ und Dämpfungswerte von $\zeta = 0,\ 0.3,\ 0.7,\ 1$. Wählen Sie dazu den Frequenzbereich $0.01 \le \omega \le 100$.

Lösungs-skizze: Die Aufgabe verdeutlicht den Einfluß der Dämpfung ζ auf den Frequenzgang des Systems. Die Lösung kann mit Hilfe von LISA ermittelt werden. Die ermittelten Kurven können abgespeichert und dann mit INGO verglichen werden (s. nachfolgende Grafik).



Aufgabe I.5: Sprungantwort eines PTn-Glieds

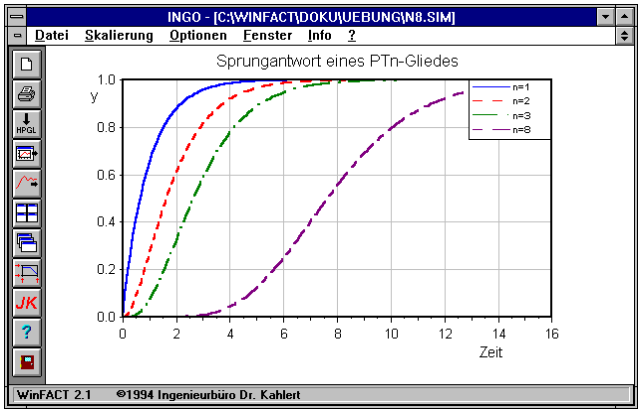
Aufgaben-stellung: Gegeben sei ein PT_n-Glied mit *n* gleichen Zeitkonstanten und der Übertragungsfunktion

$$G(s) = \frac{1}{(1+s)^n}.$$

Ermitteln Sie die Sprungantwort des Systems für *n* = 1, 2, 3 und 8. Wählen Sie eine Simulationsdauer von 15.

Lösungs-skizze: Die Simulationen lassen sich mit LISA oder BORIS (in diesem Fall einfacher!) durchführen. Die Ergebnisse können in Dateien abgelegt und dann mit INGO gegenübergestellt werden.

Man erhält folgende Verläufe:



Man erkennt, daß das System mit zunehmender Ordnung n immer "träger" wird. Für $n = 8$ tritt totzeitähnliches Verhalten auf.

Zugehörige Dateien: PTN.BSY

Aufgabe I.6: Allpaß-System

Aufgaben-stellung: Gegeben sei ein System 2. Ordnung mit der Übertragungsfunktion

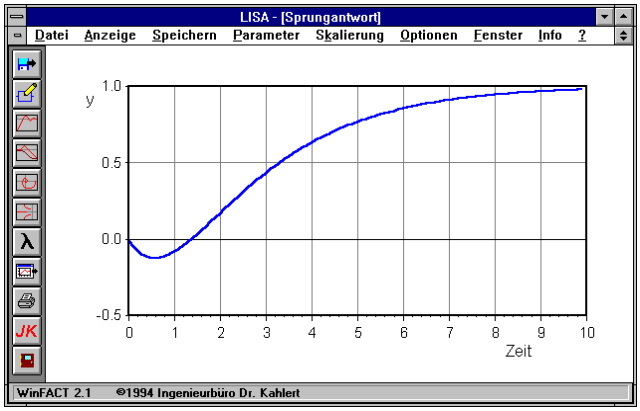
$$G(s) = \frac{1 - s}{(1 + s)(1 + 2s)}.$$

Ermitteln Sie Pole und Nullstellen des Systems. Simulieren Sie die zugehörige Sprungantwort bis zu einer Simulationszeit von 10.

Lösungs-skizze: Die Berechnung der Pol- und Nullstellen und die Durchführung der Simulation kann mit LISA erfolgen. Für Pol- und Nullstellen ergibt sich:

$$n_1 = 1 \qquad p_1 = -1 \quad p_2 = -0.5$$

Das System weist eine Nullstelle in der rechten Halbebene und damit Allpaßverhalten auf. Die Simulation ergibt das nachfolgend dargestellte Ergebnis. Man erkennt das Allpaß-typische Unterschwingen zu Beginn der Simulation.



Zugehörige Dateien: ALLPASS.UFK

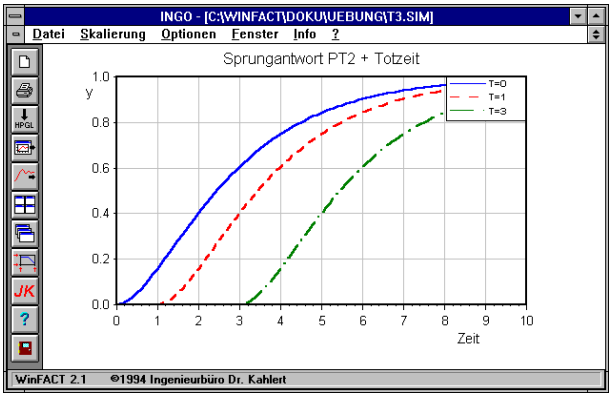
Aufgabe I.7: System mit Totzeit

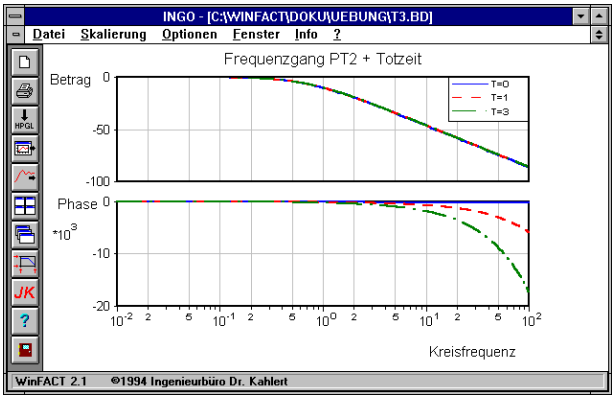
Aufgaben-stellung: Gegeben sei ein System mit der Übertragungsfunktion

$$G(s) = \frac{1}{(1+s)(1+2s)} e^{-Ts}.$$

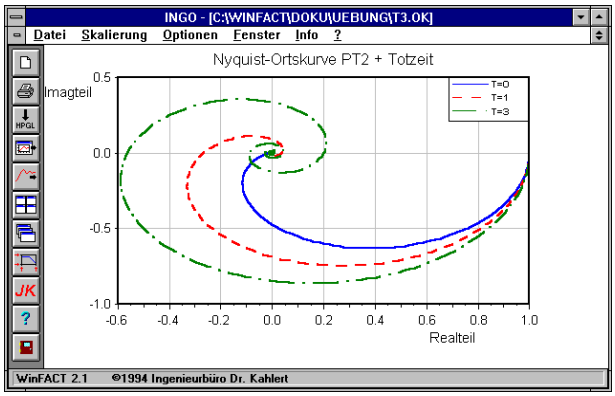
Ermitteln Sie Sprungantwort und Frequenzgang des Systems für Totzeiten T = 0, 1 und 3. Wählen Sie eine Simulationsdauer von 10 bzw. einen Frequenzbereich von $0.01 \le \omega \le 100$.

Lösungs-skizze: Diese Aufgabe verdeutlicht den Einfluß einer Totzeit auf Sprungantwort bzw. Frequenzgang eines linearen Systems. Die Berechnungen lassen sich mit LISA durchführen. Die Ergebnisse können abgespeichert und später mit INGO verglichen werden. Man erhält für Sprungantwort und Bode-Diagramm nachfolgend dargestellte Ergebnisse. Je größer die Totzeit gewählt wird, um so später setzt die Systemantwort ein. Die Totzeit hat auf die Betragskennlinie keinerlei Einfluß; sie bewirkt jedoch eine monotone Absenkung der Phasenkennlinie, die umso stärker ist, je größer die Totzeit ist.





Für den Frequenzgang in Form der Nyquist-Ortskurve erhält man schließlich:



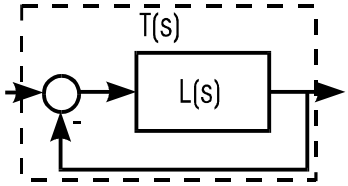
Hier macht sich die Totzeit durch das charakteristische Umkreisen des Nullpunkts bemerkbar.

Zugehörige Dateien: TOTZEIT.UFK

Kategorie II: Reglerentwurf

Aufgabe II.1: Zusammenhang zwischen Phasenreserve Φ_r und Überschwingweite M_p

Aufgabenstellung: Gegeben sei der folgende Regelkreis:



Es sei

$$L(s) = \frac{K}{s^2 + s + 1}$$

die Übertragungsfunktion des offenen Kreises. Ermitteln Sie durch Variation von *K* den Zusammenhang zwischen der Phasenreserve Φ_r des offenen Kreises $L(j\omega)$ und der daraus resultierenden Überschwingweite $M(j\omega)$ der Sprungantwort des geschlossenen Kreises

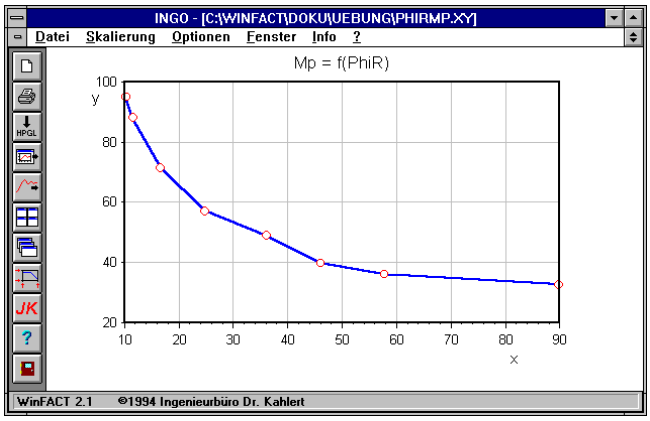
$$T(s) = \frac{L(s)}{1 + L(s)}.$$

Stellen Sie den Zusammenhang grafisch dar.

Lösungs-skizze: Zur Lösung kann das Modul RESY verwendet werden. Man erhält folgende Ergebnisse:

K	Φ_r	M_p
1	89.8	32.6
1.5	57.8	36
2	46.1	39.7
3	36.2	48.9
5	24.9	57
10	16.7	71.5
20	11.6	88
30	10.3	95

Grafisch läßt sich der Zusammenhang $M_p = f(\Phi_r)$ beispielsweise mit INGO darstellen:



Zugehörige Dateien: PHIRMP.UFK
PHIRMP.XY

Aufgabe II.2: Reglerentwurf nach Einstellregeln

Aufgaben-stellung: Gegeben sei eine Regelstrecke mit der Übertragungsfunktion

$$G(s) = \frac{0.149}{s^2 + 1.132s + 0.1772}.$$

Ermitteln Sie zunächst die Sprungantwort der Strecke bis zu einer Endzeit von 25 und lesen Sie den Verstärkungsfaktor K_S, die Verzugszeit T_u und die Ausgleichszeit T_g ab. Entwerfen Sie dann auf der Basis dieser Werte einen PID-Regler nach den Einstellregeln von Samal. Sie lauten wie folgt:

Regler mit Überschwingen der Ausgangsgröße:

$$K_R = 0.95\frac{T_g}{K_S T_u} \quad T_N = 1.35T_g \quad T_V = 0.47T_u$$

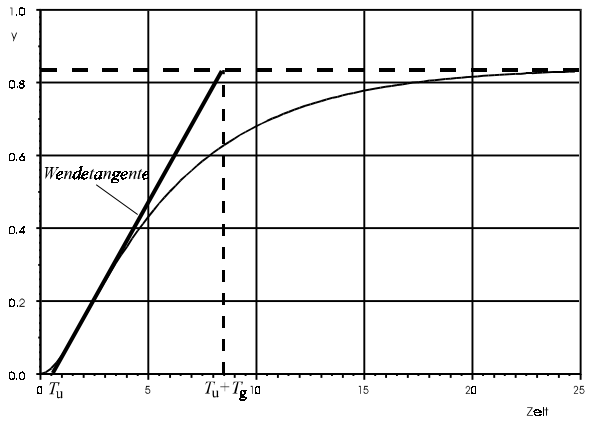
Regler ohne Überschwingen der Ausgangsgröße:

$$K_R = 0.59\frac{T_g}{K_S T_u} \quad T_N = T_g \quad T_V = 0.5T_u$$

Berechnen Sie die Sprungantworten der resultierenden Regelkreise und stellen Sie sie einander gegenüber.

Lösungsskizze: Die Berechnung der Strecken-Sprungantwort kann z. B. mit LISA erfolgen. Die Kennwerte können dann manuell bestimmt werden. Man erhält (s. auch nachfolgendes Bild)

$$K_S \approx 0.85 \quad T_u \approx 0.55 \quad T_g \approx 8$$



Daraus ergeben sich folgende Reglerparameter:

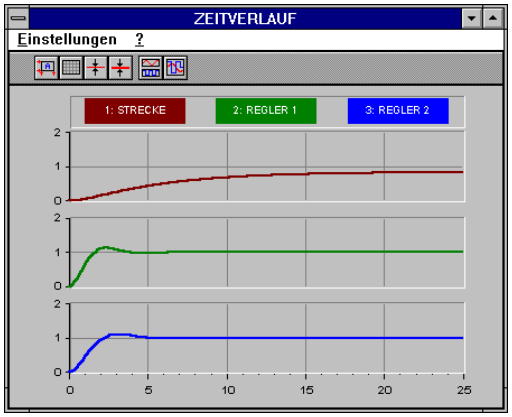
Regler mit Überschwingen der Ausgangsgröße:

$$K_R = 16.3 \quad T_N = 10.8 \quad T_V = 0.26$$

Regler ohne Überschwingen der Ausgangsgröße:

$$K_R = 10.1 \quad T_N = 8 \quad T_V = 0.28$$

Die Simulation der resultierenden Regelkreise kann z. B. mit BORIS erfolgen. Man erhält folgende Simulationsergebnisse:



Man erkennt, daß sich die Dynamik gegenüber der Strecke (obere Kurve) erheblich verbessert hat. Auch der für den überschwingfreien Fall entworfene Regler (untere Kurve) weist jedoch Überschwingen auf. Dieses ist geringer als beim ersten Regler, dafür ist jedoch auch die Anstiegszeit größer. Dieser Umstand ist im wesentlichen auf die kleinere Reglerverstärkung (10.1 im Gegensatz zu 16.3 beim ersten Regler) zurückzuführen.

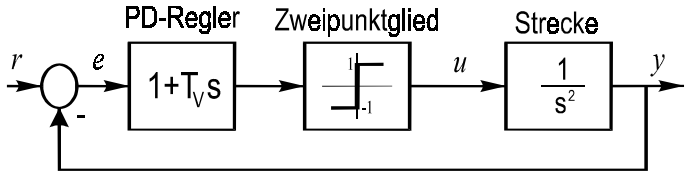
Zugehörige Dateien: EINSTELL.UFK
EINSTELL.BSY

Aufgabe II.3: Nichtlineare Regelung (Sliding-Mode-Regler)

Aufgabenstellung: Gegeben sei eine Regelstrecke der Form

$$G(s) = \frac{1}{s^2} \qquad (\textit{Doppelintegrierer}).$$

Zur Stabilisierung der Strecke soll eine Reihenschaltung aus einem PD-Regler und einem Zweipunktregler eingesetzt werden:

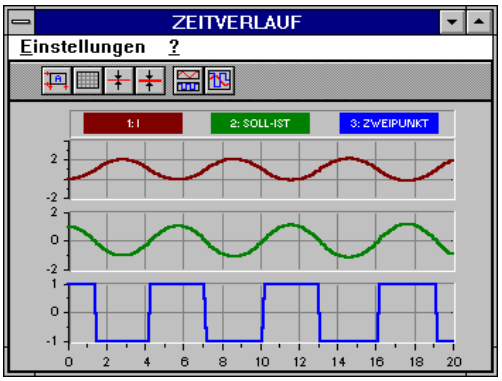


Bestimmen Sie die Vorhaltezeit T_V des PD-Reglers so, daß eine sprungförmige Eingangsgröße möglichst schnell ausgeregelt wird, die Stelleinrichtung aber nicht zu sehr belastet wird. Wählen Sie zur Simulation das Runge-Kutta-Verfahren mit der Schrittweite $\Delta T = 0.01$ und eine Simulationsdauer von 20.

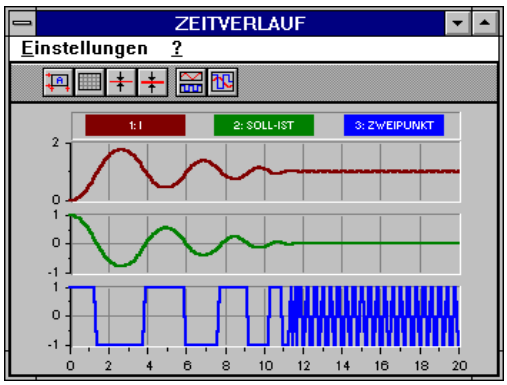
Lösungsskizze:

Bei diesem Regler handelt es sich um einen sog. *Sliding-Mode-Regler*, der die Systemtrajektorie nach anfänglicher Oszillation in einen Kriechvorgang (Sliding Mode) überführt, der dann in den stationären Zustand übergeht. Je größer die Vorhaltezeit T_V gewählt wird, umso schneller tritt der Kriechvorgang ein, aber umso länger dauert er auch. Im Kriechvorgang selbst schaltet der Zweipunktregler ständig zwischen seinen beiden Arbeitspunkten um ("Rattern"), das nachgeschaltete Stellglied wird also stark belastet. Für die Vorhaltezeit muß daher ein Kompromiß zwischen Schnelligkeit des Ausregelvorgangs und Belastung des Stellglieds gefunden werden.

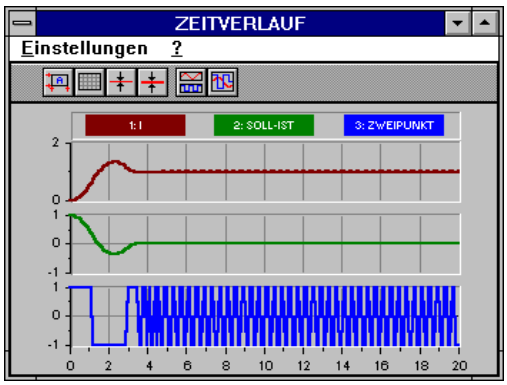
Die nachfolgenden Bilder zeigen jeweils den Verlauf der Ausgangsgröße (obere Kurve), der Regelabweichung (mittlere Kurve) und der Stellgröße (untere Kurve) für verschiedene Werte von T_V.



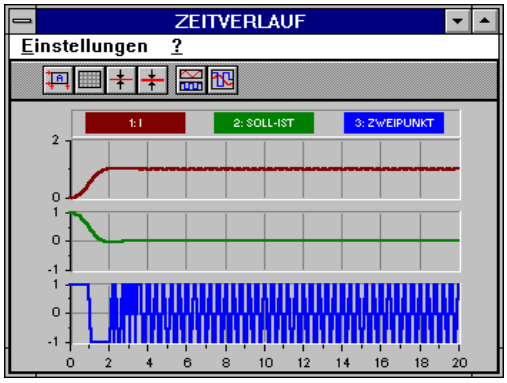
$T_V = 0$



$T_V = 0.1$



$T_V = 0.3$



$T_V = 0.5$

Für einen Wert von $T_V = 0.3$ erhält man einen brauchbaren Kompromiß.

Zugehörige Dateien: SLIDMODE.BSY

Aufgabe II.4: Reglerentwurf nach dem Frequenzkennlinienverfahren

Aufgaben-stellung: Für die Regelstrecke

$$G(s) = \frac{1}{(1+5s)(1+2s)(1+s)}$$

ist ein PID-Regler zu entwerfen, der folgende Bedingungen erfüllt:

- Durchtrittsfrequenz $\omega_c > 0.6$
- Phasenreserve $\Phi_r > 50°$
- Kreisverstärkung $V > 40$dB bei $\omega < 0.002$

Lösungs-skizze: Der Entwurf kann mit Hilfe von RESY durchgeführt werden. Zu Beginn setzen wir den P-Anteil auf den mit -1 multiplizierten Wert, den die Amplitude bei 0.6 (der Durchtrittsfrequenz) aufweist. In diesem Fall wäre das 25dB ≅ 17.78. Anschließend wird der I-Anteil so dimensioniert, daß die jetzige Amplitude (mit P-Regler) des offenen Regelkreises bei 0.002 einen Wert über 40dB (Kreisverstärkung) annimmt, damit wir später mehr Freiraum beim D-Anteil haben. In unserer Aufgabe wäre dies bei einem T_N von 17 (ca. 53 dB bei 0.002, man sollte hier jedoch nicht größer als 40dB + 25dB werden). Als letztes muß nun für die Stabilität des Kreises durch die geforderte Phasenreserve gesorgt werden. Dazu betrachten wir die Phase des offenen Regelkreises, der mittlerweile aus Strecke und PI-Regler besteht. Diese hat bei der Durchtrittsfrequenz einen um -35° von -180° verschiedenen Wert. Wir müssen die Phase also um 35° + 50° anheben, um die Forderung der Phasenreserve einzuhalten. Wir dimensionieren also unser T_V auf 8.6. Betrachtet man nun die Amplitude, so wird man feststellen, daß diese noch nicht bei 0.6 durchtritt. Aus diesem Grund wird der P-Anteil auf 1 gesetzt und wie oben beschrieben neu bestimmt. Danach müßten die gewünschten Werte erreicht sein.

Die Parameter des Reglers haben letztendlich folgendes Aussehen

$$K_R = 3.5 \quad T_N = 17 \quad T_V = 8.6$$

Zugehörige Dateien: FKL.UFK

Aufgabe II.5: Wurzelortskurve

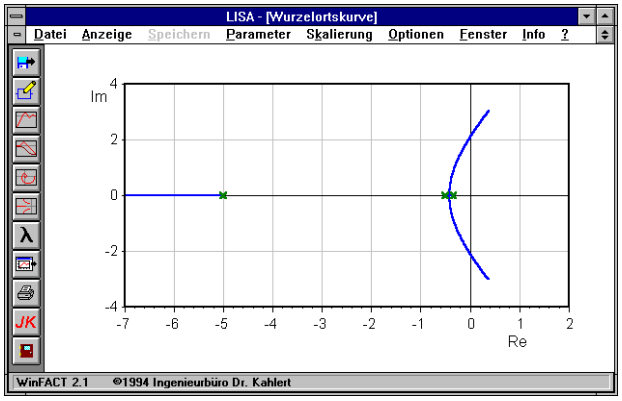
Aufgaben-stellung: Gegeben sei die Regelstrecke

$$G(s) = \frac{1}{(1+0.2s)(1+2s)(1+3s)}.$$

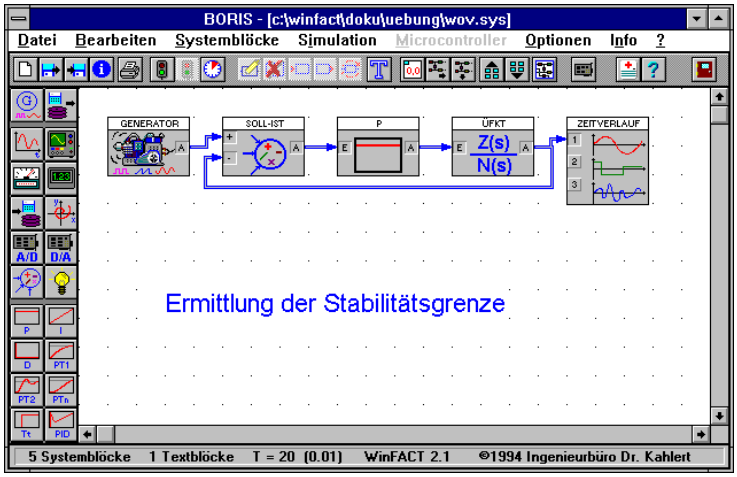
Die Strecke soll durch einen P-Regler zu einem geschlossenen Regelkreis ergänzt werden.

Berechnen Sie zunächst die zugehörige Wurzelortskurve. Ermitteln Sie dann simulatorisch die Reglerverstärkung K_R, bei der der geschlossene Regelkeis instabil wird.

Lösungs-skizze: Die Wurzelortskurve kann mit Hilfe von LISA erzeugt werden. Sie hat folgende Gestalt:



Die Simulation kann mit Hilfe von BORIS durchgeführt werden. Dazu muß eine möglichst kleine Schrittweite, z. B. $\Delta T = 0.01$, in Verbindung mit dem Runge-Kutta-Verfahren gewählt werden, um im Bereich der Stabilitätsgrenze noch hinreichend genaue Ergebnisse zu erhalten.



Die Simulation ergibt, daß der Regelkreis bei einer Reglerverstärkung von $K_R \approx 29$ instabil wird. Dieser Wert läßt sich z. B. mit Hilfe des Nyquist-Kriteriums anhand der Ortskurve verifizieren.

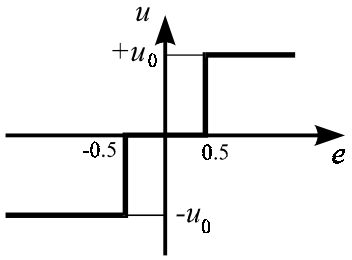
Zugehörige Dateien: WOV.UFK
WOV.BSY

Aufgabe II.6: Nichtlinearer Regelkreis mit Dauerschwingung

Aufgaben-stellung: Gegeben sei ein Einfachregelkreis mit der Regelstrecke

$$G(s) = \frac{1}{(1+0.5s)^3}$$

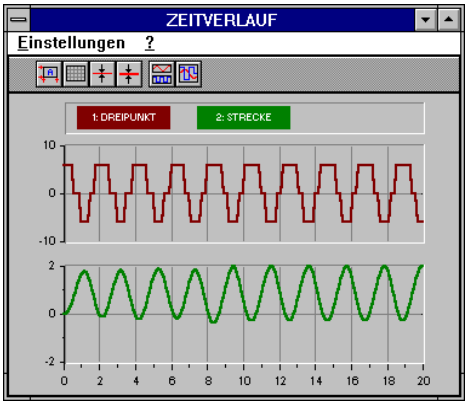
und dem nachfolgenden Dreipunktregler:



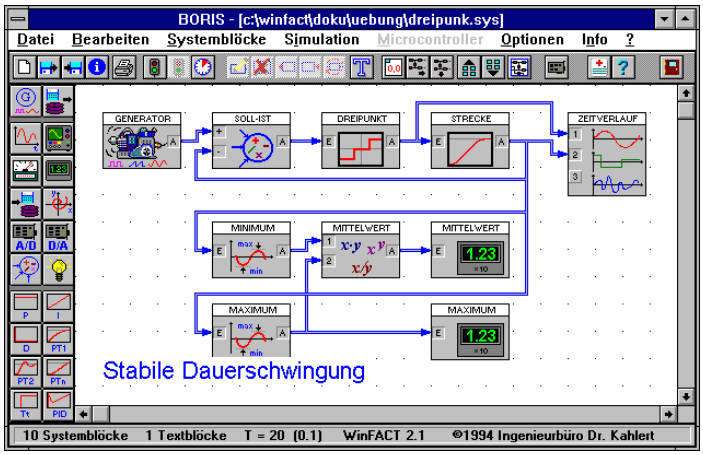
Bestimmen Sie mit BORIS die Stellgröße u_0 so, daß die sich einstellende Dauerschwingung bei einem Eingangssprung einen Maximalwert von 2 nicht überschreitet, der Mittelwert der Dauerschwingung aber mindestens bei 0.8 liegt.

Lösungsskizze:

Die Amplitude der sich einstellenden Dauerschwingung steigt mit der Stellgröße u_0. Durch Bestimmung von Minimum und Maximum der Dauerschwingung kann ein Schätzwert für den Mittelwert ermittelt werden, der zweckmäßigerweise über ein Digitalinstrument angezeigt wird. Die Stellgröße u_0 wird dann schrittweise erhöht, bis die Bedingungen erfüllt sind. Für $u_0 = 5.9$ ergeben sich die gewünschten Werte (obere Kurve: Stellgröße; untere Kurve: Ausgangsgröße):



Das Simulationssystem hat dabei folgende Struktur:



Zugehörige Dateien: DREIPUNK.BSY

Kategorie III: Simulation

Aufgabe III.1: Räuber-Beute-System ohne Kapazitätsbegrenzung [10]

Aufgabenstellung: Das dynamische Verhalten einer *Beutepopulation x* und einer *Räuberpopulation y* wird beschrieben durch das Dgl.-System

$$\dot{x} = a\,x - b\,x\,y$$
$$\dot{y} = c\,x\,y - d\,y$$

Darin ist

- *a:* spezifische Wachstumsrate der Beutepopulation
- *b:* spezifische Beuteverlustrate der Beute
- *c:* spezifische Beutegewinnrate der Räuberpopulation
- *d:* spezifische Atmungsrate der Räuberpopulation

Die Interaktion zwischen Räuber und Beute führt zu Verlusten bei der Beutepopulation und Gewinnen für die Räuberpopulation. Wird trotz Vermehrung der Beute diese zu stark dezimiert, so reduziert dies auch die Energiezufuhr der Räuber und damit ihren Bestand. Das "Beutemachen" hängt sowohl von der Beutepopulation *x* wie von der Räuberpopulation *y* ab. Die proportionale Abhängigkeit von beiden Größen führt zur Nichtlinearität *xy*. Entsprechende Verluste werden bei der Beute abgezogen (Term *bxy*) und bei der Räuberpopulation als Gewinn gebucht (Term *cxy*). Die Verluste bei der Beutepopulation werden durch deren bestandsproportionalen Zuwachs mit der spezifischen Zuwachsrate *a* teilweise wettgemacht. Der Räuber hingegen braucht die Beute zur

Kompensation seiner normalen Atmungsverluste (Parameter d) und damit zur Lebenserhaltung.

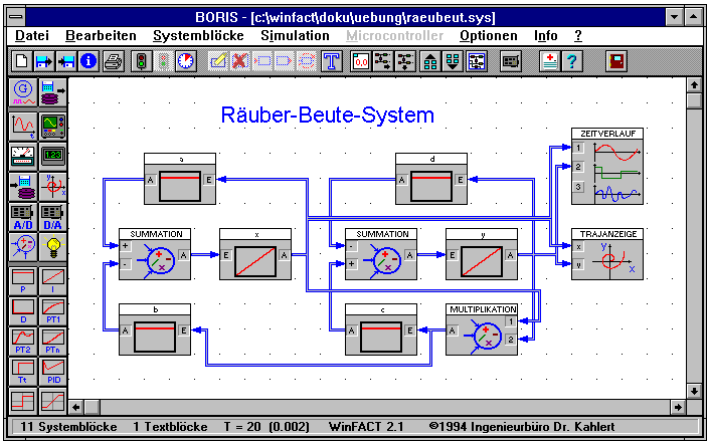
Simulieren Sie mit Hilfe von BORIS das dynamische Verhalten des Systems für folgende Parameter:

$$a = b = c = d = 1$$
$$x(t=0) = y(t=0) = 0.1$$
$$T_{\text{Simu}} = 20, \Delta T = 0.002$$

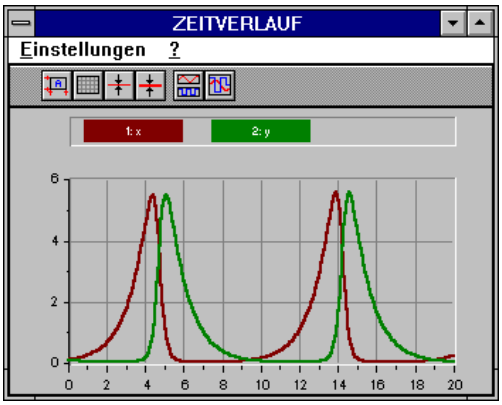
Ermitteln Sie die Zeitverläufe $x(t), y(t)$ sowie die Trajektorie $y(x)$.

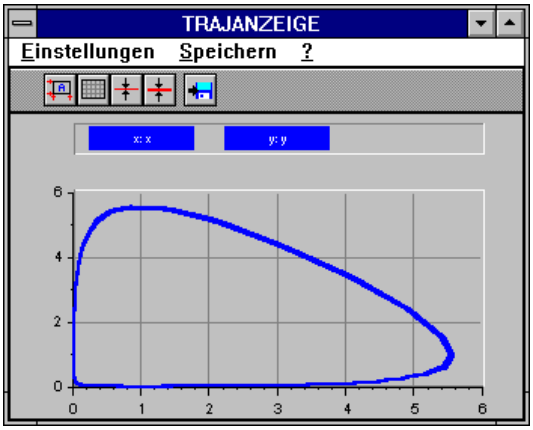
Lösungs-skizze:

Das System weist folgende Struktur auf:



Es ergeben sich für die gewählten Parameter die folgenden Verläufe:





Zugehörige Dateien: RAEUBEUT.BSY

Aufgabe III.2: Tourismus und Umwelt [10]

Aufgabenstellung: Eine wegen ihrer natürlichen Umwelt besonders attraktive Region zieht Touristen an. Die natürliche Umwelt kann sich durch Regeneration bis an eine Kapazitätsgrenze wieder erneuern, sie wird aber durch den Tourismus belastet und teilweise zerstört. Dadurch sinkt ihre ursprüngliche Attraktivität und der Tourismus geht zurück. Diese Zusammenhänge werden beschrieben durch das Dgl.-System

$$\dot{x} = -a\,x + b\,y$$
$$\dot{y} = d\,y(1 - y/k) - c\,y$$

Darin ist

x:	normierte Touristenzahl
y:	Umweltqualität
a:	spezifische Touristenverlustrate
b:	Werbewirkung
c:	spezifische Rate der Umweltzerstörung
d:	spezifische Rate der Umwelterholung
k:	Tragfähigkeit der Umwelt

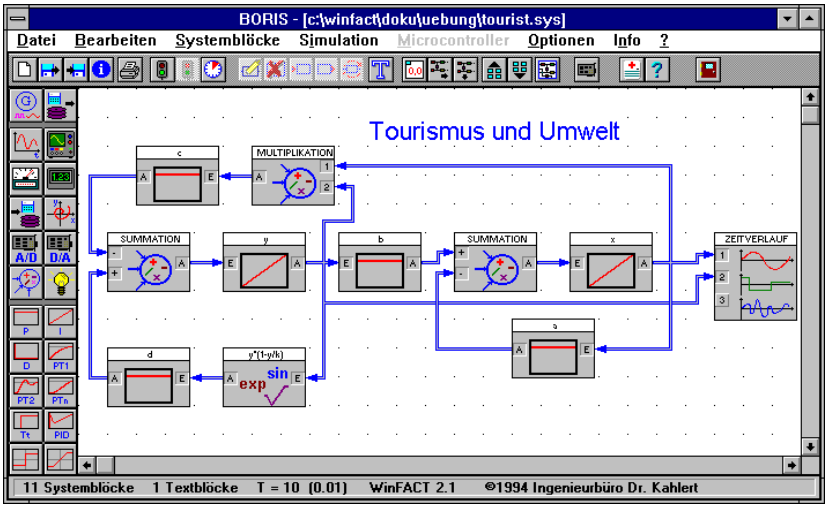
Die Attraktivität einer Region hängt von ihrem Umweltzustand ab und kann noch durch Werbung (Parameter *b*) verstärkt werden. Der Touristenzustrom ist proportional dieser Attraktivität (Term *by*) und erhöht die Touristenpopulation *x*. Diese hat ständige Verluste (Parameter *a*) durch abreisende Touristen. Die Umweltbeeinträchtigung durch die Touristen hängt von der Touristenzahl und dem Umweltzustand selbst ab und ist daher proportional zu *xy* und einer spezifischen Belastungsrate *c*. Alleingelassen würde die Umwelt sich nach anfänglicher Beeinträchtigung wieder bis an die Kapazitätsgrenze *k* mit einer spezifischen Regenerationsrate *d* regenerieren.

Ermitteln Sie die Zeitverläufe $x(t)$, $y(t)$ mit BORIS für folgende Parameter:

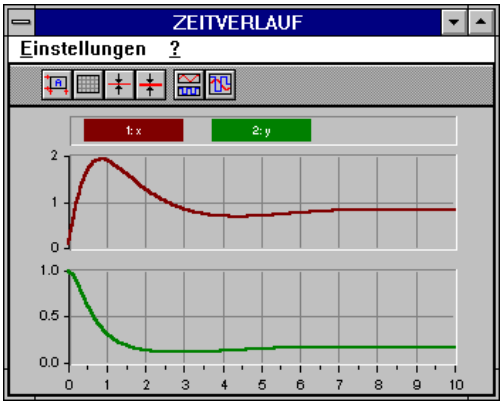
$$a = c = d = k = 1 \qquad b = 5$$
$$x(t=0) = 0.1 \quad y(t=0) = 1$$
$$T_{\text{Simu}} = 10, \;\; \Delta T = 0.01$$

Lösungsskizze:

Das System weist folgende Struktur auf:



Es ergeben sich folgende Ergebnisse:



Zugehörige Dateien: TOURIST.BSY

Aufgabe III.3: Rotationspendel [10]

Aufgaben-stellung: Ein masseloser, starrer Stab der Länge *r* ist in einem Endpunkt drehbar gelagert, so daß er in einer senkrechten Ebene rotieren kann. Am anderen Ende des Stabs befindet sich eine punktförmige Masse *m*. Wird das Pendel anfangs mit hoher Winkelgeschwindigkeit *y* angestoßen, so kann es mehrfach um den Drehpunkt rotieren, bevor es schließlich nach einigem mit der Dämpfung *d* gedämpften Hin- und Herpendeln am unteren Totpunkt zum Stillstand kommt. Das System wird beschrieben durch das Dgl.-System

$$\dot{x} = y$$

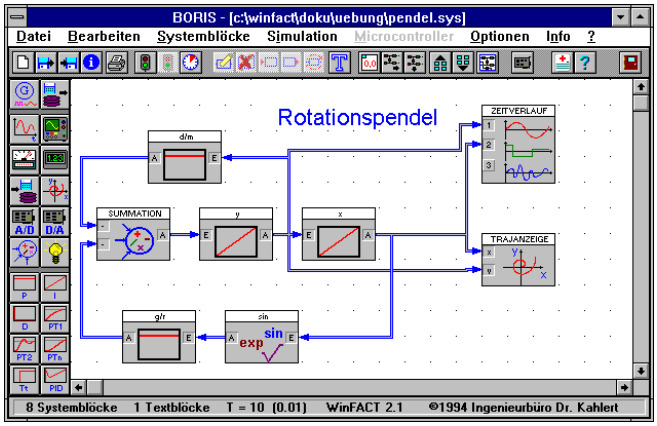
$$\dot{y} = -\frac{g}{r}\sin x - \frac{d}{m}y \qquad g = 9.81\,\mathrm{m/s^2}$$

x ist die Winkelauslenkung des Pendels.

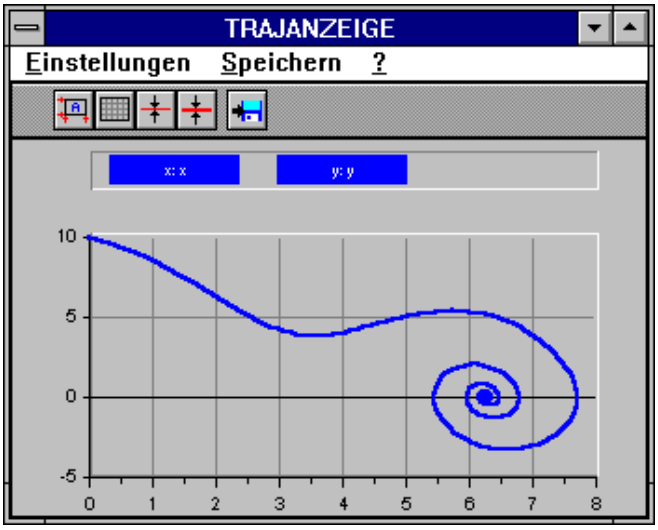
Ermitteln Sie mit Hilfe von BORIS den Trajektorienverlauf $y(x)$ für folgende Parameter:

$$m = r = d = 1$$
$$x(t=0) = 0$$
$$y(t=0) = 10$$
$$T_{\mathrm{Simu}} = 10, \quad \Delta T = 0.01$$

Lösungs-skizze: Das System weist folgende Simulationsstruktur auf:



Man erhält die nachfolgenden Ergebnisse:



Zugehörige Dateien: PENDEL.BSY

Aufgabe III.4: Chaotischer bistabiler Schwinger [10]

Aufgaben-stellung:

Das System besteht im Kern aus einem linearen Schwinger, bei dem jedoch eine negative Rückkopplung von x^3 auf y besteht. Das System wird sinusförmig erregt. Es ergibt sich auf diese Weise eine chaotische Bewegung. Das zugehörige Dgl.-System lautet

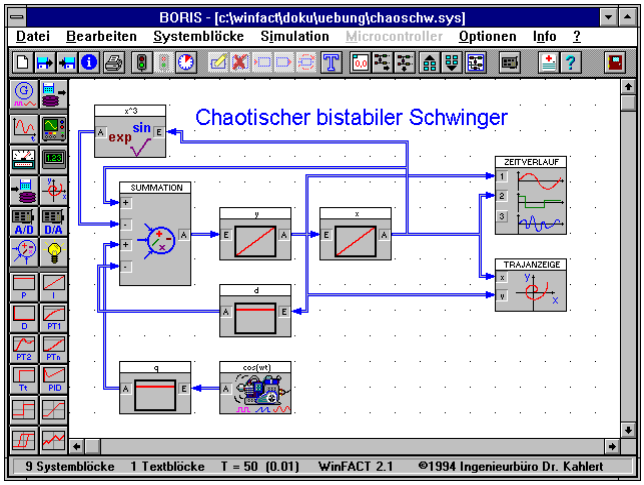
$$\dot{x} = y$$
$$\dot{y} = x - x^3 - d\,x + q\cos\omega t$$

Ermitteln Sie die Zeitverläufe $x(t)$, $y(t)$ sowie den Trajektorienverlauf für folgende Parameter:

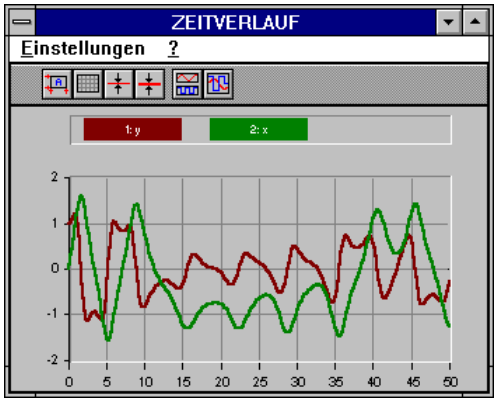
$$d = 0.25 \quad q = 0.3 \quad \omega = 1$$
$$x(t=0) = 0 \quad y(t=0) = 1$$
$$T_{\text{Simu}} = 50, \quad \Delta T = 0.01$$

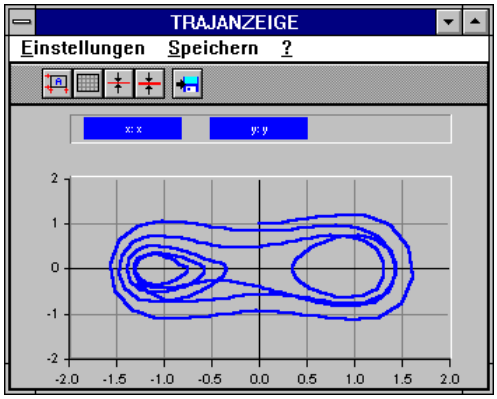
Lösungs-sikzze:

Das Simulationssystem weist die folgende Struktur auf:



Es ergeben sich folgende Simulationsergebnisse:





Zugehörige Dateien: CHAOSCHW.BSY

Aufgabe III.5: Lorenz-System [10]

Aufgabenstellung: Das Lorenz-System ist eine angenäherte Darstellung der hydro-thermodynamischen Grundgleichungen für die Kopplung von Wärmekonvektion und Wärmeleitung bei Flüssigkeitsströmungen. Die Zustandsgröße *x* beschreibt das Geschwindigkeitsprofil, die Zustandsgrößen *y* und *z* die Temperaturverteilung. Liegen die Parameter in einem gewissen Bereich, so weist das System chaotisches Verhalten auf.

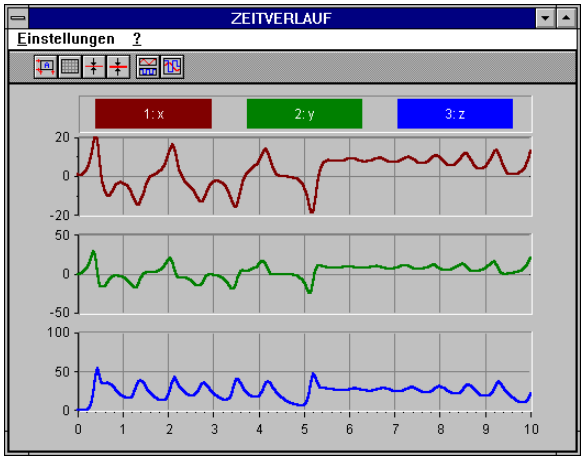
Das Dgl.-System lautet

$$\dot{x} = a(y - x)$$
$$\dot{y} = -xz + bx - y$$
$$\dot{z} = xy - cz$$

Ermitteln Sie mit Hilfe von BORIS die Zeitverläufe $x(t), y(t), z(t)$ für:

$$a = 10, b = 28, c = 2.667$$
$$x(t = 0) = 1,\ y(t = 0) = z(t = 0) = 0$$
$$T_{\text{Simu}} = 10, \Delta T = 0.01$$

Lösungs-skizze: Es ergeben sich folgende Simulationsergebnisse:



Zugehörige Dateien: LORENZ.BSY

Aufgabe III.6: Verkoppelte Dynamos [10]

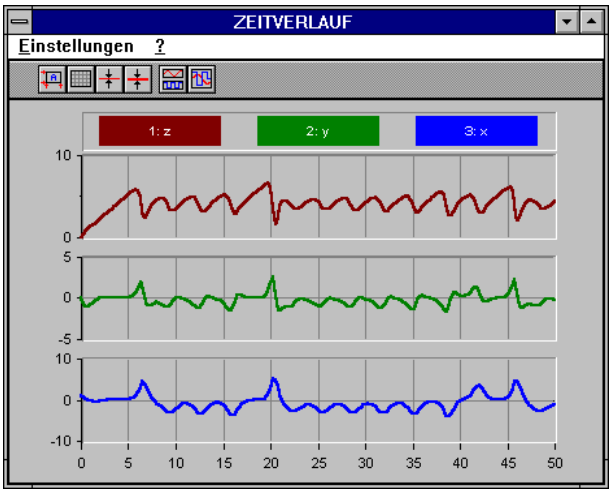
Aufgaben-stellung: Zwei identische Dynamos sind miteinander gekoppelt, wobei der Strom des einen Dynamos jeweils das magnetische Feld des anderen erregt. Die Ströme in den beiden Stromkreisen sind die Zustandsgrößen *x* und *y*. Die Zustandsgröße *z* ist die Rotationsgeschwindigkeit für den Dynamo *x*. Der Parameter *c* gibt die Differenz der Rotationsgeschwindigkeiten beider Dynamos an. Das System weist chaotisches Verhalten auf. Das Dgl.-System lautet

$$\dot{x} = z\,y - a\,x$$
$$\dot{y} = (z-c)x - a\,y \qquad c = a(b^2 - 1/b^2)$$
$$\dot{z} = 1 - x\,y$$

Ermitteln Sie die Verläufe der drei Zustandsgrößen für folgende Parameter:

$$a = 1, \quad b = 2$$
$$x(t=0) = 1$$
$$y(t=0) = z(t=0) = 0$$
$$T_{\text{Simu}} = 50, \quad \Delta T = 0.01$$

Lösungs-skizze: Man erhält folgende Simulationsergebnisse:



Zugehörige Dateien: DYNAMOS.BSY

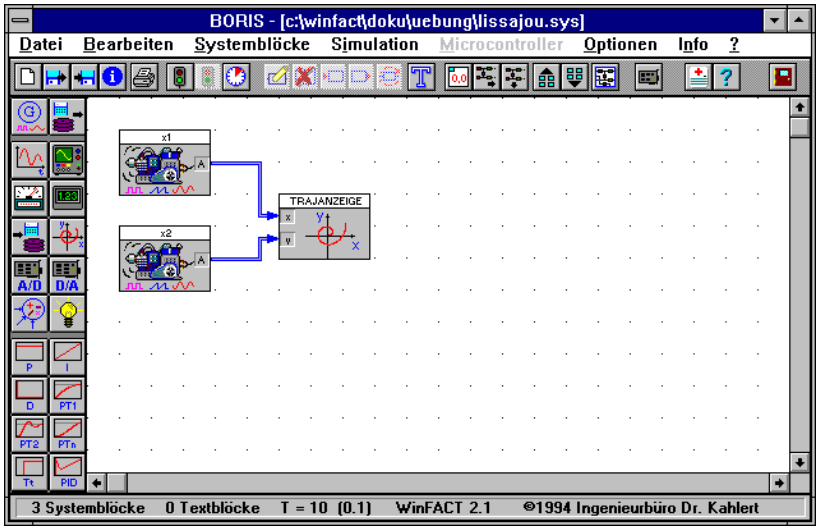
Aufgabe III.7: Lissajous-Figuren

Aufgaben-stellung: Gegeben seien die beiden Zeitfunktionen

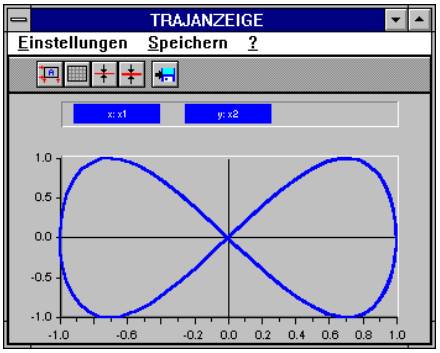
$$x_1(t) = \sin t$$
$$x_2(t) = \sin n\,t$$

Ermitteln Sie die Trajektorienverläufe $x_2(x_1)$ für n = 2, 4 und 6 und eine Simulationsdauer von 10.

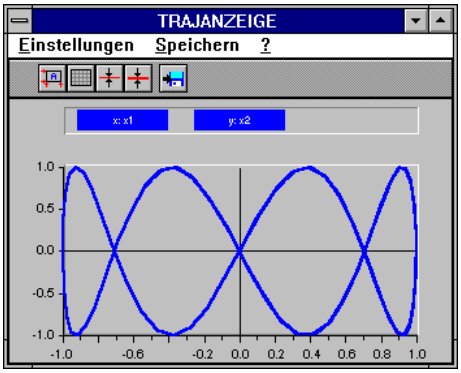
Lösungs-skizze: Zur Lösung mit Hilfe von BORIS werden zwei Generatoren, die als Sinusgeneratoren betrieben werden, sowie eine Trajektorienanzeige benötigt.



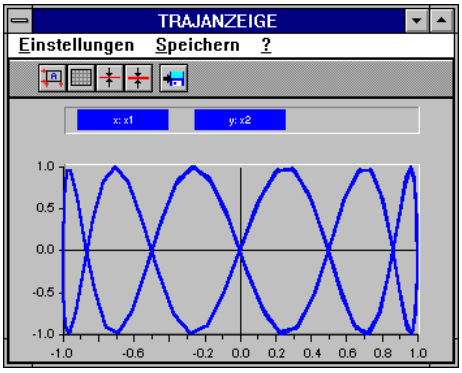
Je nach Wahl von n erhält man die bekannten Lissajous-Figuren:



n = 2



n = 4



n = 6

Zugehörige Dateien: LISSAJOU.BSY

Aufgabe III.8: Van der Pol - Differentialgleichung [12]

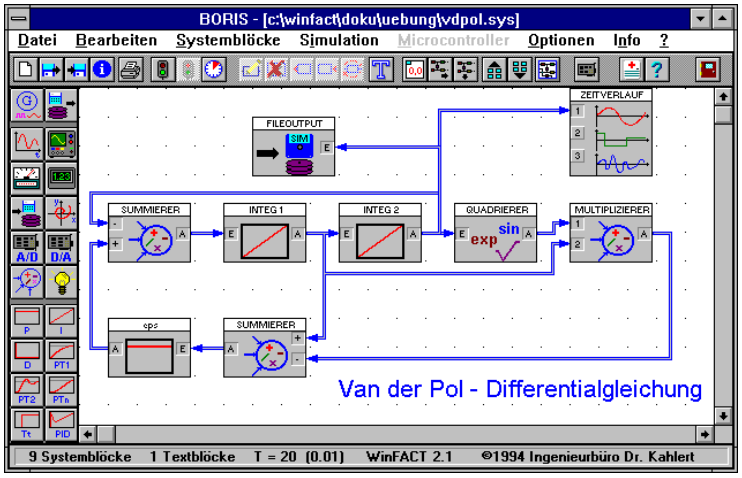
Aufgaben-stellung: Gegeben sei die Van der Pol - Differentialgleichung

$$\ddot{y} + \varepsilon(y^2 - 1)\dot{y} + y = 0$$

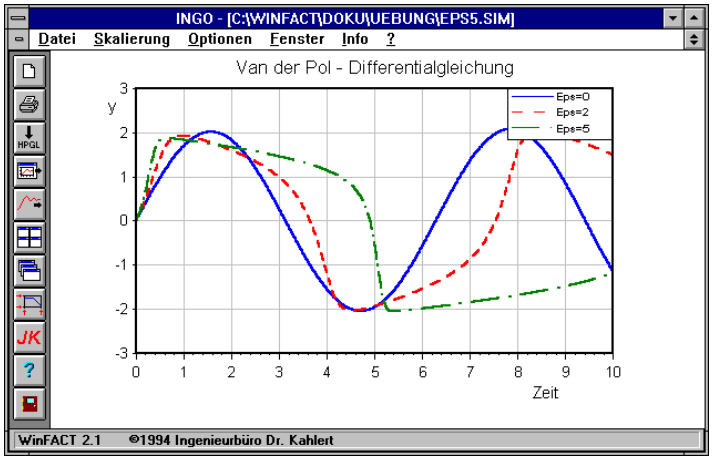
Berechnen Sie mit Hilfe von BORIS die Lösung der Dgl. für folgende Parameter:

$$y(t=0)=2 \quad \dot{y}(t=0)=0$$
$$\varepsilon = 0,\ 2,\ 5$$
$$T_{\text{Simu}} = 20,\ \Delta T = 0.01$$

Lösungs-skizze: Das System ist zweiter Ordnung und weist folgende Struktur auf:



Die Simulationsergebnisse für die unterschiedlichen Werte von ε werden zweckmäßigerweise in einer Datei abgespeichert und können dann mit INGO verglichen werden. Man erhält folgende Ergebnisse:



Der Wert des Parameters ε ist maßgeblich verantwortlich für das nichtlineare Verhalten des Systems. Für den Sonderfall ε = 0 erhält man einen linearen Oszillator.

Zugehörige Dateien: VDPOL.BSY

Aufgabe III.9: Fast-Fourier-Transformation

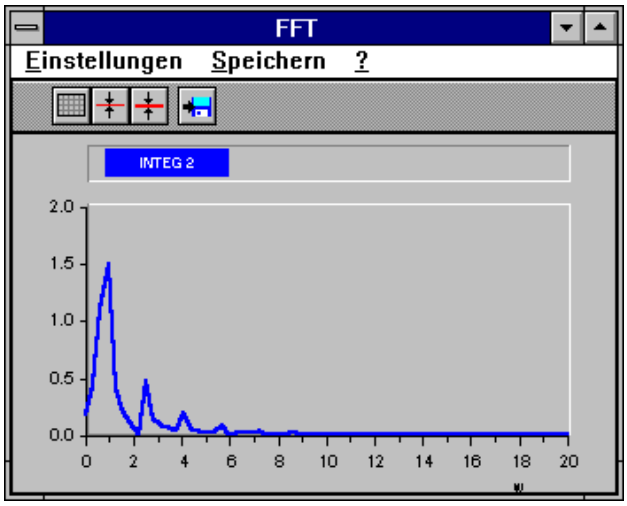
Aufgaben-stellung: Gegeben sei die Van der Pol - Differentialgleichung

$$\ddot{y} + \varepsilon(y^2 - 1)\dot{y} + y = 0$$

Berechnen Sie mit Hilfe von BORIS das Amplitudenspektrum von $y(t)$ für folgende Parameter:

$$y(t=0) = 2 \quad \dot{y}(t=0) = 0$$
$$\varepsilon = 0,\ 2,\ 5$$
$$T_{\text{Simu}} = 20,\ \Delta T = 0.01$$

Lösungs-skizze: Die aus Aufgabe II.8 bekannte Struktur wird ergänzt durch einen FFT-Block. Man erhält folgendes Ergebnis:



Zugehörige Dateien: FFT.BSY

Aufgabe III.10: Stabilität von Integrationsverfahren [12]

Aufgaben-stellung: Gegeben sei ein PT_1-Glied mit der Übertragungsfunktion

$$G(s) = \frac{K}{1+Ts}$$

und den Parametern

$$K = 1,\ T = 2$$

Simulieren Sie das Verhalten des freien Systems, d. h. für den Fall $u(t) \equiv 0$, bei einer Anfangsauslenkung von $y(0) = 1$ für verschiedene Simulationsschrittweiten ΔT

- mit dem Euler-Verfahren
- mit dem Runge-Kutta-Verfahren.

Vergleichen Sie die erhaltenen Ergebnisse mit der exakten Lösung

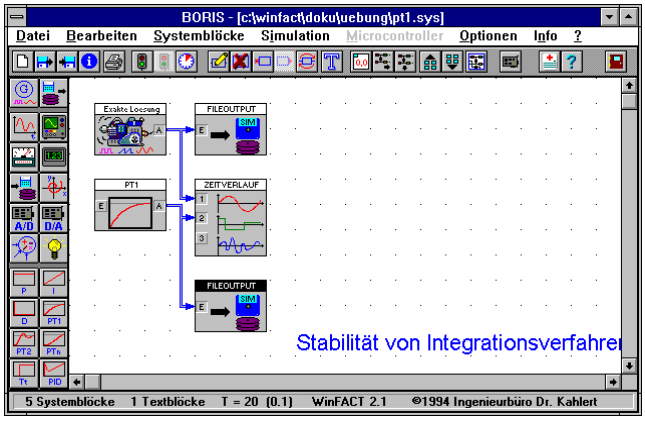
$$y(t) = e^{-t/2}.$$

Bei welcher Simulationsschrittweite werden die Integrationsverfahren instabil?

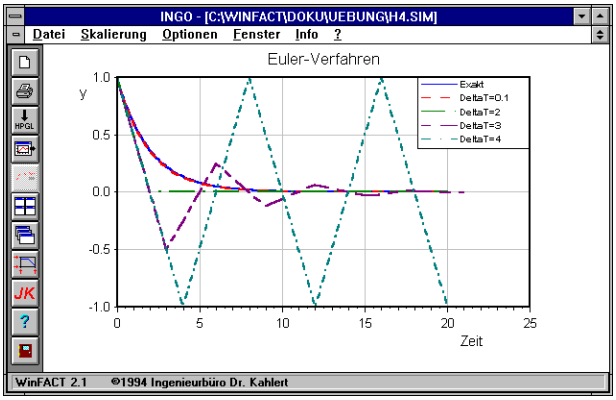
Lösungs-skizze: Diese Aufgabe soll verdeutlichen, welchen Einfluß die Simulationsschrittweite auf die Genauigkeit des Simulationsergebnisses hat und wie geeignete Schrittweiten in Abhängigkeit von der Dynamik des simulierten Systems zu finden sind. In der Regel gilt, daß die Schrittweite bei linearen Systemen maximal 1/10 der kleinsten Systemzeitkonstanten - in diesem Beispiel also 1/10 von 2 - betragen sollte.

Die Simulation kann mit Hilfe von BORIS durchgeführt werden. Dabei kann der Eingang des PT_1-Blocks offen bleiben. Die Ergebnisse können in Dateien abgelegt werden und dann mit INGO verglichen werden.

Die nachfolgende Grafik zeigt zunächst die Simulationsstruktur:

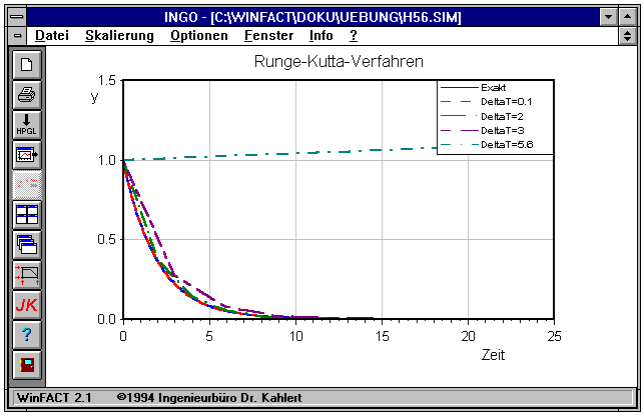


Zur Ermittlung der exakten Lösung kann ein Generator benutzt werden, der über den Funktionsparser programmiert wird. Die nachfolgende Grafik zeigt die Simulationsergebnisse, die mit Hilfe des Euler-Verfahrens ermittelt wurden. Dabei wurden Simulationsschrittweiten von $\Delta T = 0.1,\ 2,\ 3$ und 4 benutzt.



Man erkennt, daß das Simulationsergebnis für die Schrittweite von 0.1 mit der exakten Lösung nahezu übereinstimmt. Diese Schrittweite beträgt 1/20 der Systemzeitkonstanten und ist daher ausreichend klein. Bei Vergrößerung von ΔT weicht das Simulationsergebnis immer mehr von der exakten Lösung ab. Bei $\Delta T = 4$ schließlich wird das Euler-Verfahren instabil.

Die Ergebnisse für das Runge-Kutta-Verfahren zeigt nachfolgende Grafik. Man erkennt, daß dieses Verfahren erst bei einer Schrittweite von etwa 5.6 instabil wird.



Zugehörige Dateien: PT1.BSY

Aufgabe III.11: Steife Systeme

Aufgaben-stellung: Gegeben sei ein System mit der Übertragungsfunktion

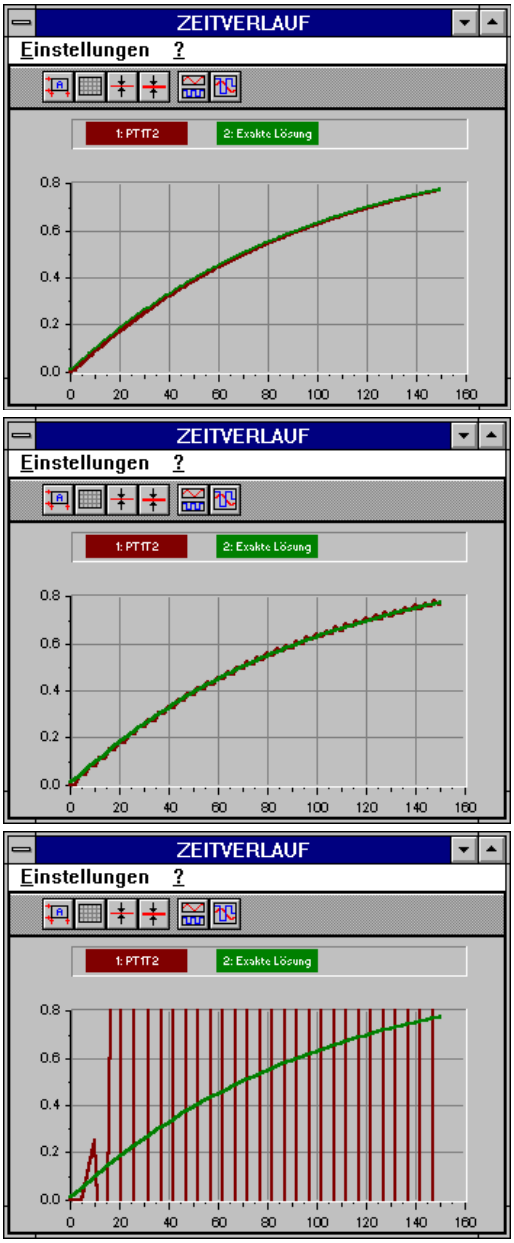
$$G(s) = \frac{1}{(1+s)(1+100s)}.$$

Berechnen Sie die Sprungantwort des Systems mit Hilfe des Euler-Verfahrens für die Simulationsschrittweiten ΔT =0.1, 2, 5 bis zu einer Endzeit von 150 und vergleichen Sie die erhaltenen Ergebnisse mit der exakten Lösung

$$y(t) = 1 + \frac{1}{99} e^{-t} - \frac{100}{99} e^{-t/100}.$$

Lösungs-skizze: Bei dem gegebenen System handelt es sich um ein steifes System mit den Zeitkonstanten $T_1 = 1$ und $T_2 = 100$. Wählt man eine sehr kleine Schrittweite, so ist die Simulation zwar hinreichend genau, aber die Simulationsdauer muß extrem lang gewählt werden, um die langsame Eigenbewegung zu erfassen. Wählt man dagegen die Simulationsschrittweite groß, so wird die Simulation ungenau bzw. sogar instabil.

Man erhält folgende Simulationsergebnisse:



Ergebnisse für $\Delta T = 0.1$ *(oben),* $\Delta T = 2$ *(mitte),* $\Delta T = 5$ *(unten)*

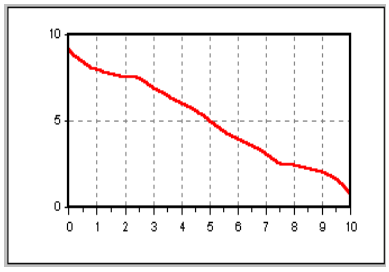
Zugehörige Dateien: STEIFSYS.BSY

Kategorie IV: Fuzzy-Logik

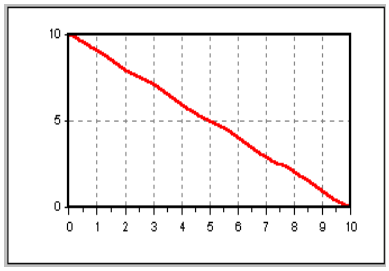
Aufgabe IV.1: Kennlinien-Generierung

Aufgaben-stellung: Definieren Sie eine Eingangsvariable und eine Ausgangsvariable mit jeweils fünf Sets. Teilen Sie die Sets gleichförmig auf den Bereich von null bis zehn auf, indem Sie die Standardform wählen. Versuchen Sie, folgende Kennlinien nachzubilden:

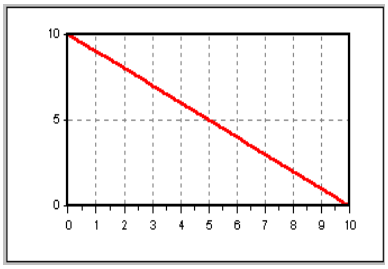
a)



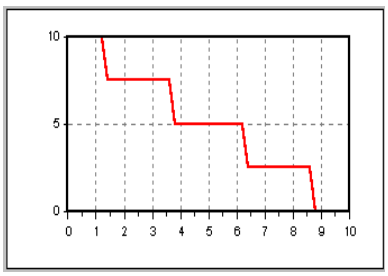
b)



c)



d)



Lösungs-skizze:

a) Schwerpunktmethode (Max-Prod-Inferenz)

Bei der unter a) abgebildeten Kennlinie werden die Endpunkte (0,10) und (10,0) nicht erreicht. Der Kurvenverlauf weist an den Enden leichte Biegungen auf, ist sonst aber recht glatt. Die Biegungen resultieren aus den Unsymmetrien der Randsets.

b) Modifizierte Schwerpunktmethode (Max-Min-Inferenz)

Kennlinie b) erreicht die Endpunkte. Die an den Endpunkten unter a) befindlichen Biegungen sind hier nicht mehr vorhanden, da die Randsets bei dieser Methode symmetrisch erweitert werden.

c) Näherungsweise Schwerpunktmethode

Da alle Sets bei der Schwerpunktsberechnung als Singletons betrachtet werden und diese gleichmäßig über den Wertebereich der linguistischen Variablen verteilt sind, ist der Kurvenverlauf eine Gerade von einem zum anderen Endpunkt.

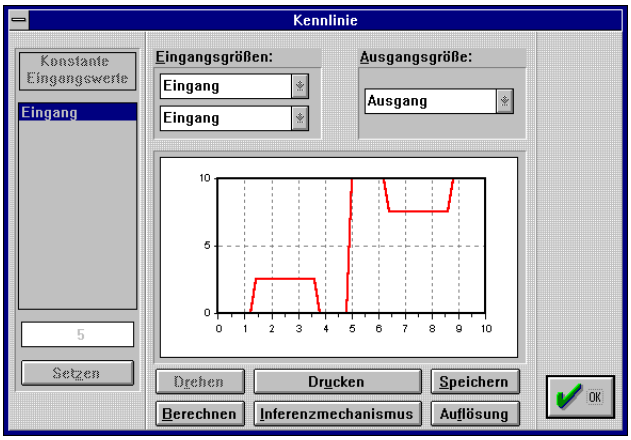
d) Maximale Höhe (rechts/links)

Die Kurve verläuft sprungförmig, da immer nur das Ausgangsset der Regel mit dem höchsten Erfülltheitsgrad zur Defuzzifizierung herangezogen wird.

Zugehörige Dateien: DEFUZZY.FUZ

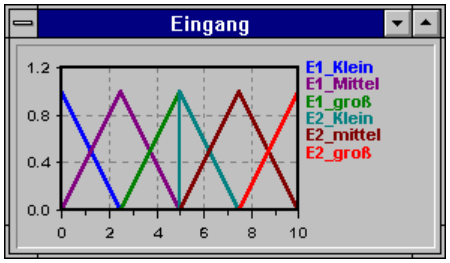
Aufgabe IV.2: Fuzzy-Logik mit einem Eingang und einem Ausgang

Aufgaben-stellung: Es gilt folgendes Problem zu lösen: Sie haben nur noch einen Eingang auf Ihrem µC-Board für Fuzzy-Logik frei. Verwenden Sie diesen Eingang so, daß es möglich ist, zwei unterschiedliche Signale getrennnt voneinander zu verarbeiten. Definieren Sie in Ihrer linguistischen Eingangsvariablen sechs Sets von Hand und in Ihrer Ausgangsvariablen vier Sets in Standardform. Überlegen Sie, wie die Eingangs-Sets, die Regelbasis und die Defuzzifizierung aussehen müssen, wenn die folgende Kennlinie gewünscht wird:

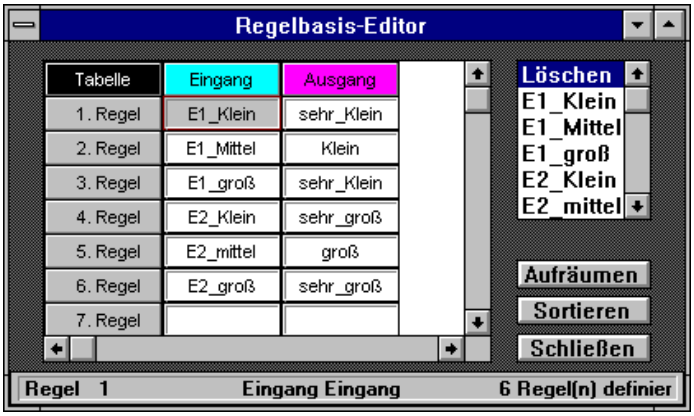


Wie müssen die einzelnen Signale verarbeitet werden, bevor sie in den Fuzzy-Baustein geleitet werden?

Lösungs-skizze: Die Eingangssets sind folgendermaßen zu definieren:

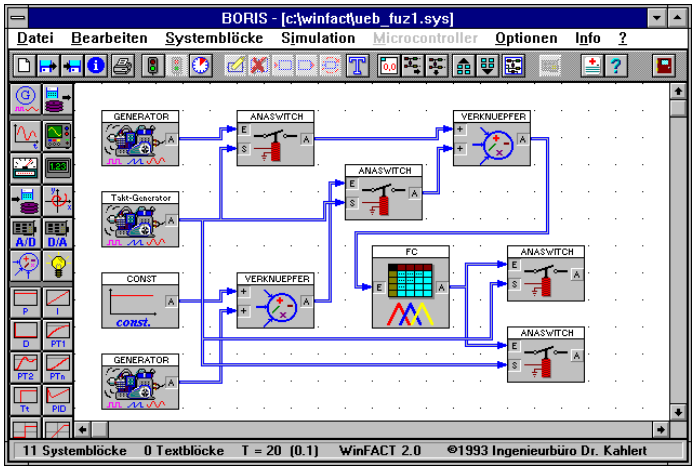


Deutlich zu sehen ist, daß die Mitte des Wertebereiches von jeweils einem Set berührt wird. Betrachtet man nur die untere bzw. obere Hälfte des Wertebereiches, so läßt sich wieder die sonst übliche Konfiguration erkennen. Die Regelbasis dürfte nun schnell ersichtlich sein:



Da die Kennlinie nur horizontale Bereiche aufweist (der endlich steile Sprung in der Mitte des Definitionsbereichs ist auf die begrenzte Auflösung der Kennlinie zurückzuführen), wird es sich bei der Defuzzifizierung um eine der Maximalen Höhe handeln (siehe Aufgabe IV.1 d).

Problematischer hingegen ist die Signalaufbereitung vor dem Fuzzy-Block. Nach Definition der linguistischen Eingangsvariablen sollten die beiden Eingangssignale unterschiedliche Amplitudenbereiche verwenden: Das erste Signal den Bereich von null bis fünf, das andere Signal den Bereich von fünf bis zehn. Sind beide Eingangssignale von gleicher Änderungsgeschwindigkeit, so erfolgt die Abtastung abwechselnd mit genügend hoher Frequenz. Ansonsten muß über einen Frequenzteiler das gewünschte Abtastverhältnis eingestellt werden. Die Eingangssignalverarbeitung kann also etwa wie folgt aussehen:

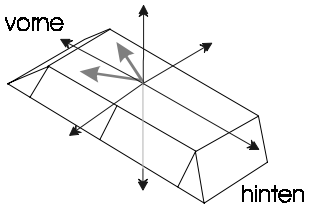


Die Ausgangssignale der beiden Relais können dann in geeigneter Weise weiterverarbeitet werden.

Zugehörige Dateien: ZWEI_EIN.FUZ
ZWEI_EIN.BSY

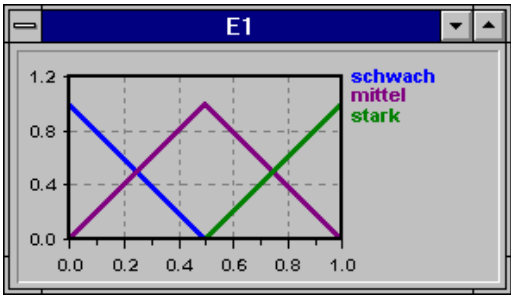
Aufgabe IV.3: Fuzzy-Logik mit zwei Eingängen

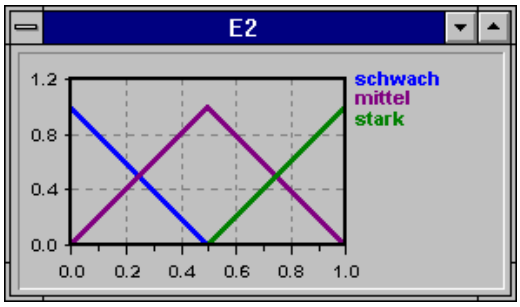
Aufgabenstellung: Ein ferngesteuertes Solarspielzeugauto soll, sobald die Ladung der Akkus zur Neige geht, von selbst in eine beleuchtete Position fahren, um seine Akkumulatoren wieder aufzuladen. Das Auto hat zwei lichtempfindliche Transistoren, die auf dem Autodach in folgender Weise angebracht sind (graue Pfeile):

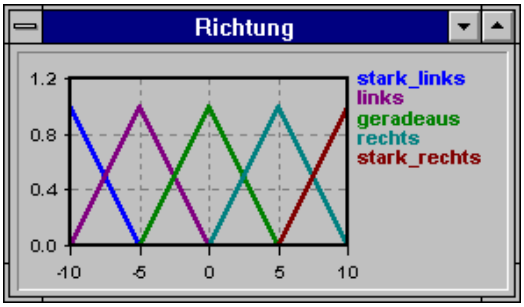


Stellen Sie eine Fuzzy-Logik auf, die die Lichtsuche steuert. Wählen Sie für die Eingänge drei Sets im Bereich von null bis eins, für den Ausgang fünf Sets im Bereich von -10 bis 10. Überlegen Sie, welche Defuzzifizierungsmethode am geeignetsten ist, wenn Sie den kompletten Wertebereich der Ausgangsvariablen ausnutzen wollen.

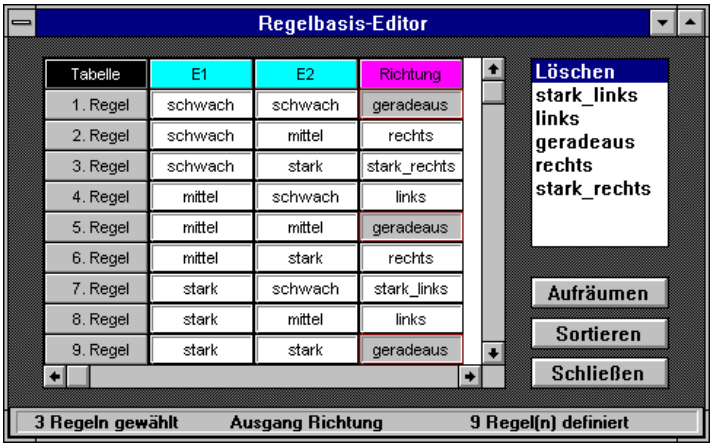
Lösungs-skizze: Die linguistischen Variablen können in Standardform angesetzt werden:



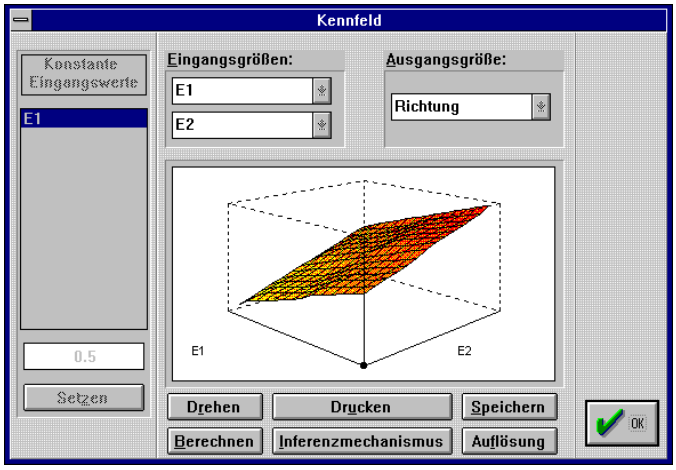




Die Regelbasis sollte so beschaffen sein, daß immer dann als Richtung geradeaus erfolgt, wenn die Eingänge beide das gleiche Set ansprechen. Es ergibt sich folgende Regelbasis:



Da der gesamte Wertebereich der Ausgangsvariablen ausgeschöpft werden soll und ein möglichst gleichförmiger Kennfeldverlauf erwünscht ist (schnelle Fahrtrichtungswechsel könnten das Auto zum Schleudern bringen), stehen zwei Defuzzifizierungsmethoden zur Verfügung: die näherungsweise Schwerpunktmethode und die modifizierte Schwerpunktmethode. Die schnellere der beiden - also die näherungsweise Schwerpunktmethode - wird gewählt. Der Controller weist dann folgendes Kennfeld auf:

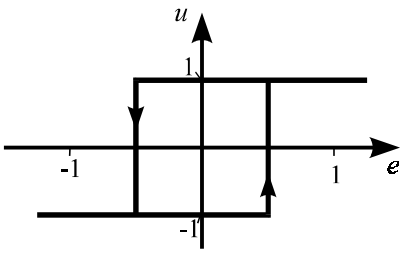


Zugehörige Dateien: AUTOPARK.FUZ

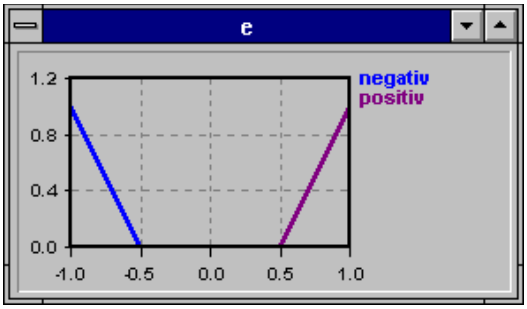
Kategorie V: Fuzzy-Control

Aufgabe V.1: Fuzzy-Controller mit Hysterese [2]

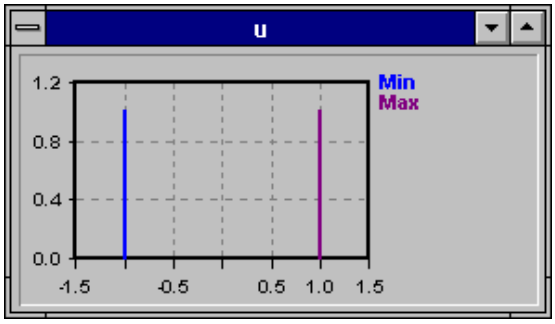
Aufgaben-stellung: Entwerfen Sie einen Fuzzy-Controller mit der Eingangsgröße $e(t)$ und der Ausgangsgröße $u(t)$, der die folgende Kennlinie mit Hysterese aufweist:



Lösungs-skizze: Um die Hysterese zu erzeugen, darf im Eingangsgrößenbereich $-0.5 \le e \le 0.5$ keine Regel aktiv sein. Der Fuzzy-Controller muß in diesem Fall die zuletzt ermittelte Stellgröße u beibehalten. Dies kann man z. B. dadurch erreichen, daß man nur zwei Fuzzy-Sets für e definiert, zwischen denen eine "Lücke" im Bereich der Hysterese ist. Die Form der Fuzzy-Sets spielt keine Rolle:

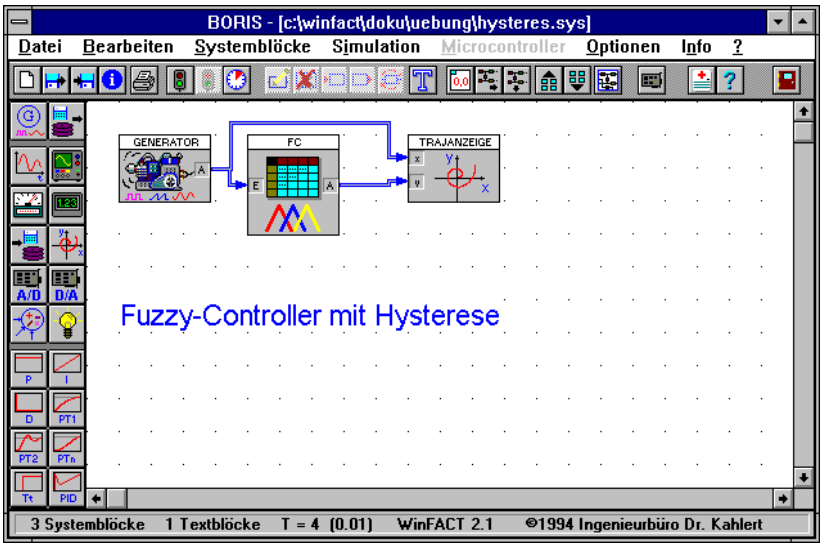


Für die Fuzzy-Sets der Stellgröße u können wir Singletons an der Stelle -1 bzw. 1 wählen:

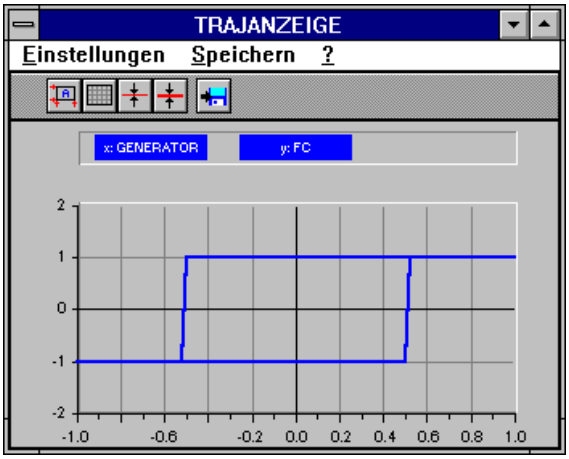


Inferenzmechanismus und Defuzzifizierungsmethode können wir beliebig wählen, da ohnehin immer nur maximal eine Regel aktiv ist.

Zum Austesten unseres Fuzzy-Controllers müssen wir BORIS benutzen (die Fuzzy-Shell FLOP kann keine Kennlinien mit Hysterese darstellen). Dazu legen wir an unseren Fuzzy-Controller eine zunächst linear von -1 auf 1 ansteigende und dann wieder auf -1 abfallende Eingangsgröße an. Ein- und Ausgangsgröße des Fuzzy-Controllers geben wir auf eine Trajektorienanzeige:



Wir erhalten den folgenden Verlauf:



Zugehörige Dateien: HYSTERES.FUZ
HYSTERES.BSY

Aufgabe V.2: Vergleich: Konventioneller P-Regler und Fuzzy-P-Regler [2]

Aufgaben-stellung: Gegeben sei eine Reglerstrecke mit der Übertragungsfunktion

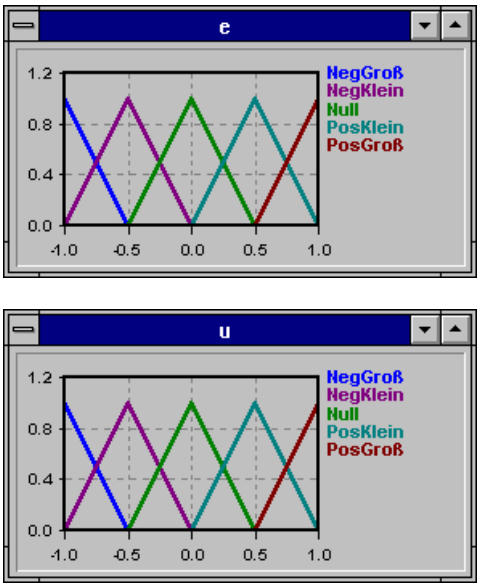
$$G(s) = \frac{1}{(1+s)(1+2s)}.$$

Entwerfen Sie für diese Strecke einen Fuzzy-P-Regler mit jeweils fünf Fuzzy-Sets für Regelabweichung und Stellgröße in Standardform und dem Wertebereich [-1, 1]. Simulieren Sie die Sprungantwort des resultierenden Regelkreises mit Hilfe von BORIS bis zu einer Endzeit von 10 für folgende Fälle:

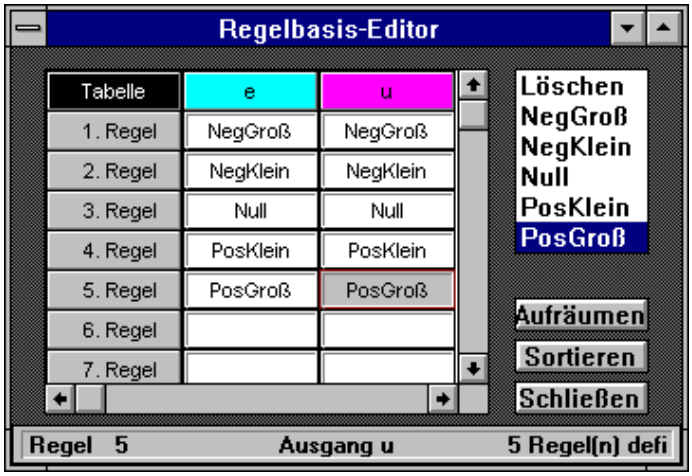
- Max-Min-Inferenz und modifizierte Schwerpunktmethode
- Max-Prod-Inferenz und modifizierte Schwerpunktmethode
- Max-Min-Inferenz und Maximum-Methode

Vergleichen Sie die Ergebnisse mit denen eines konventionellen P-Reglers mit der Verstärkung 1.

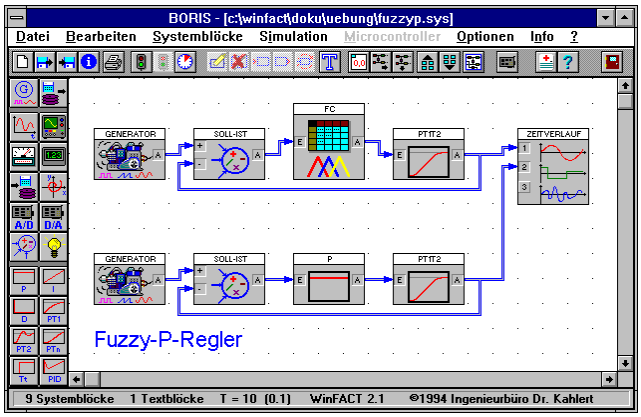
Lösungs-skizze: Der Fuzzy-P-Regler kann mit Hilfe von FLOP entworfen werden. Die Fuzzy-Sets haben folgende Gestalt:



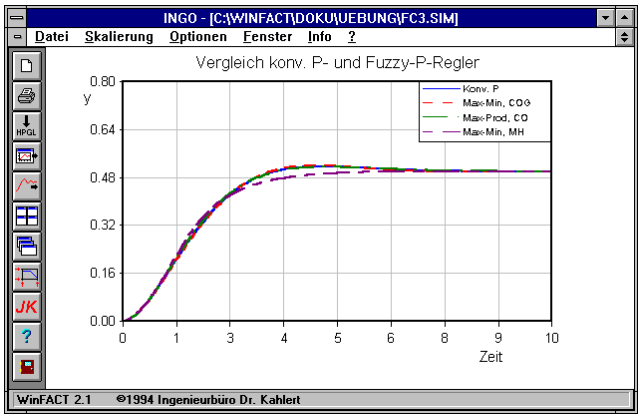
Die Regelbasis weist folgende Struktur auf:



Die Simulation wird mit Hilfe von BORIS vorgenommen. Dabei werden beide Regelkreise (mit Fuzzy-P-Regler und mit konventionellem P-Regler) parallel simuliert:



Die Ergebnisse werden abgespeichert und können dann mit INGO verglichen werden. Man erhält folgende Verläufe:

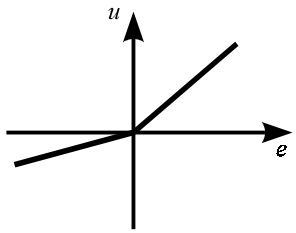


Man erkennt deutlich, daß die Fuzzy-Regler mit der Defuzzifizierung nach der Schwerpunktmethode sowohl bei Max-Min- als auch bei Max-Prod-Inferenz nahezu das gleiche Regelverhalten aufweisen wie der konventionelle P-Regler. Lediglich bei Defuzzifizierung nach der Maximum-Methode erhält man schlechtere Ergebnisse, da der Fuzzy-P-Regler in diesem Fall eine stufenförmige Kennlinie (Multirelais-Charakteristik) aufweist.

Zugehörige Dateien: FUZZYP.FUZ
FUZZYP.BSY

Aufgabe V.3: Fuzzy-Controller als Split-Range-Regler

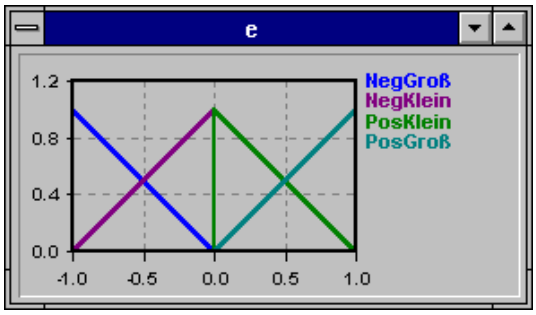
Aufgabenstellung: Entwerfen Sie einen Fuzzy-Controller mit der Eingangsgröße *e* und der Ausgangsgröße *u*, der unterschiedliche Verstärkungen für positive und negative Eingangsgrößen aufweist, also eine Kennlinie folgender Form besitzt:



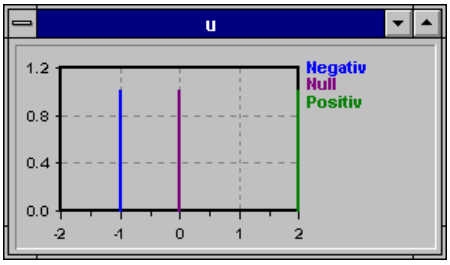
Die Verstärkung für negative Eingangsgrößen soll 1, die für positive Eingangsgrößen 2 betragen.

Lösungsskizze: Eine der möglichen Lösungen sieht folgendermaßen aus:

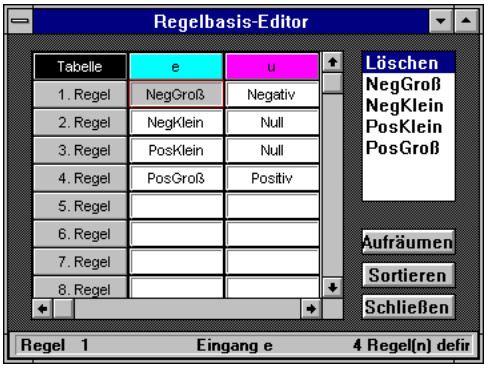
Die Fuzzy-Sets für *e* werden wie folgt gewählt:



Für die Stellgröße *u* werden Singletons gewählt, die an den Stellen -1, 0 und 2 liegen:



Die Regelbasis schließlich hat folgende Struktur:



Als Inferenzart wird Max-Min-Inferenz und zur Defuzzifizierung die Schwerpunktmethode für Singletons gewählt.

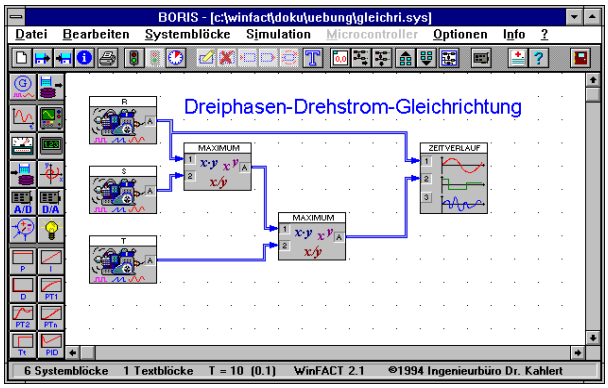
Zugehörige Dateien: SPLITRAN.FUZ

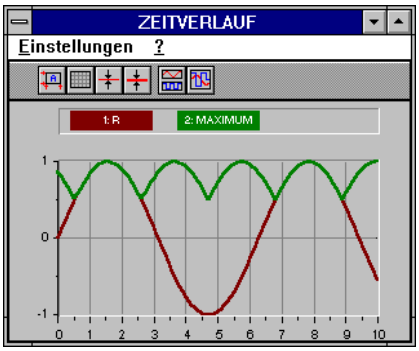
Kategorie VI: Meßtechnik

Aufgabe VI.1: Dreiphasen-Drehstrom-Gleichrichtung

Aufgaben-stellung: Geben Sie eine Schaltung zur Einweggleichrichtung eines Dreiphasen-Drehstroms an.

Lösungs-skizze: Die Gleichrichtung der drei Phasen kann über zwei Funktionsblöcke vorgenommen werden, die jeweils das Maximum zweier Eingangsgrößen bestimmen. Die nachfolgenden Bilder verdeutlichen die Struktur und den resultierenden Signalverlauf.



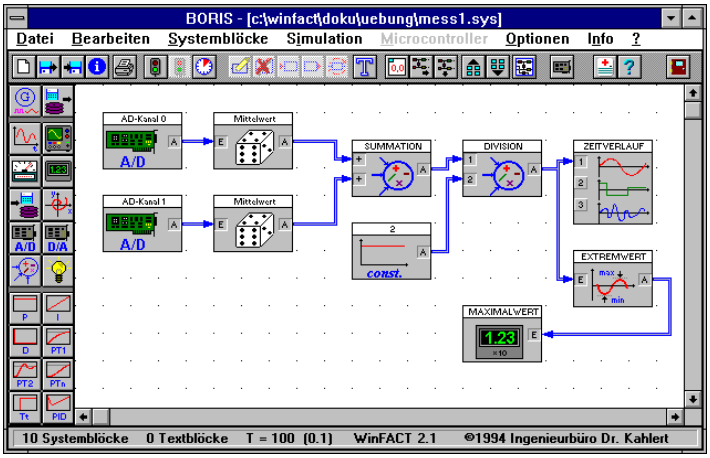


Zugehörige Dateien: GLEICHRI.BSY

Aufgabe VI.2: Meßtechnische Erfassung von Signalen

Aufgaben-stellung: Lesen Sie zwei Signale über die A/D-Wandlerkarte ein und bestimmen Sie deren Mittelwert. Geben Sie den Mittelwert als Funktion über der Zeit aus. Zusätzlich soll das bisherige Maximum auf einer Digitalanzeige sichtbar gemacht werden.

Lösungs-skizze: Folgender Schaltungsaufbau löst die Aufgabe:

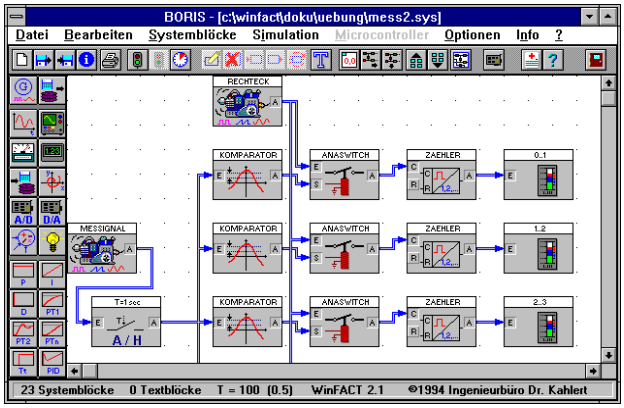


Zugehörige Dateien: MESS1.BSY

Aufgabe VI.3: Statistische Auswertung durch Balkendiagramme

Aufgaben-stellung: Ein stochastisches analoges Signal soll mit der Abtastzeit $\Delta T = 1$ abgetastet werden. Man weiß, daß dieses Signal im Durchschnitt alle Werte von 0 bis 5 mit unterschiedlicher, aber bekannter Häufigkeit durchläuft. Verschaffen Sie sich einen Überblick, ob das Signal korrekt abgetastet wird, indem Sie das Signal in fünf gleichgroße Klassen einteilen und sich die Anzahl der Elemente der Klassen über Balkendiagramme anzeigen lassen. Realisieren Sie dazu eine Schaltung, die ohne Integrierer auskommt.

Lösungs-skizze: Die Klasseneinteilung erfolgt durch Komparatoren, die wiederum an Vorwärtszähler angeschlossen sind. Da die Zähler flankengetriggert sind, müssen die statischen Komparatorpegel über ein Relais und einen als Rechteckgenerator betriebenen Generator in Impulse umgewandelt werden. Damit die Schaltung ordnungsgemäß arbeitet, darf die Simulationsschrittweite höchstens die Hälfte der gewählten Abtastzeit - hier also maximal 0.5 - betragen.



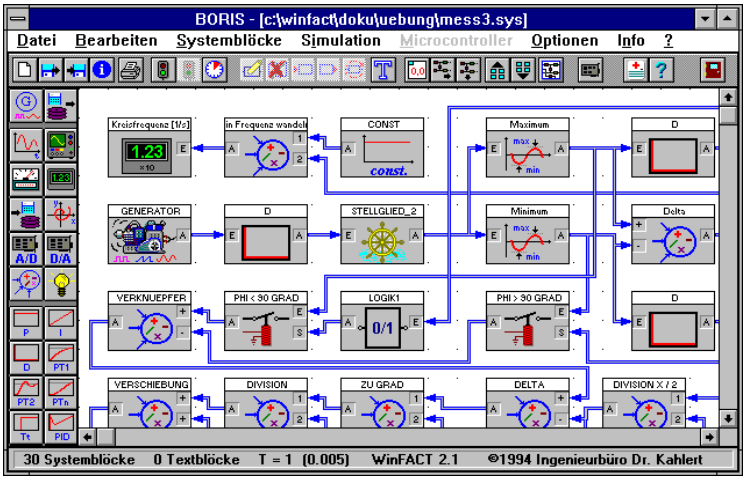
Die angegebene Struktur ist nur eine mögliche Lösung. Eine andere Lösung, die mit weniger Blöcken auskommt, besteht darin, auf die Relais zu verzichten und stattdessen das abgetastete Signal hinter dem A/H-Glied mit dem Signal des Rechteckgenerators zu multiplizieren. Der Rechteckgenerator muß in diesem Fall allerdings die Amplitude 1 aufweisen.

Zugehörige Dateien: MESS2.BSY

Aufgabe VI.4: Messung von Phase und Frequenz

Aufgaben-stellung: Es ist ohne weiteres möglich, sich die Frequenz und die Phasenverschiebung eines Sinussignals über den Oszillographen anzusehen. Der Nachteil dabei ist, daß die Zeitauflösung des Oszillographen stets neu gesetzt werden muß, sobald Sie ein Eingangssignal mit anderer Frequenz anlegen. In diesem Beispiel soll versucht werden, die Frequenz und die Phasenverschiebung auf den Digitalanzeigen sichtbar zu machen. So entfällt das Einstellen des Oszillographen und man kann die Werte unmittelbar ablesen. Entwerfen Sie eine Schaltung, die diese Aufgabe erledigt.

Lösungs-skizze: Durch das Inkrementieren bzw. Dekrementieren während der positiven bzw. negativen Halbwelle (Stellglied_2) erhalten wir einen Maximal- und Minimalwert, in denen die komplette benötigte Information vorhanden ist. Der Offsetwert dieser Dreiecksfunktion kann zur Phasenverschiebung, der Spitze-Spitze-Wert zur Frequenzumrechnung herangezogen werden. Nachfolgende Grafik zeigt einen Ausschnitt der Schaltung:



Zugehörige Dateien: MESS3.BSY

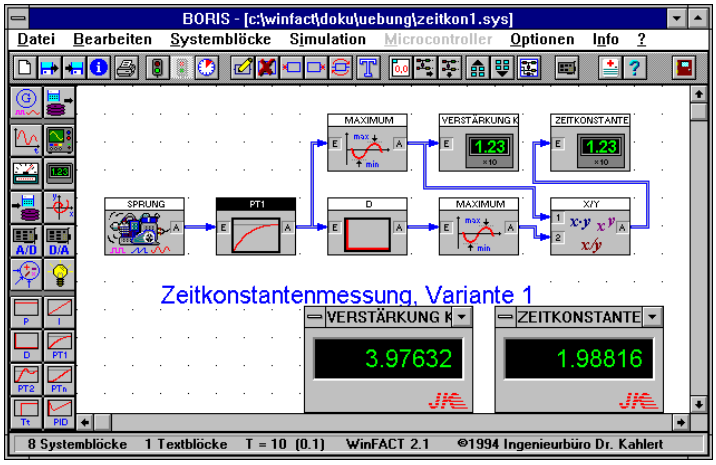
Aufgabe VI.5: Meßtechnische Bestimmung der PT_1-Zeitkonstante

Aufgabenstellung: Gesucht ist eine Schaltung zur meßtechnischen Bestimmung der Zeitkonstante eines PT_1-Glieds

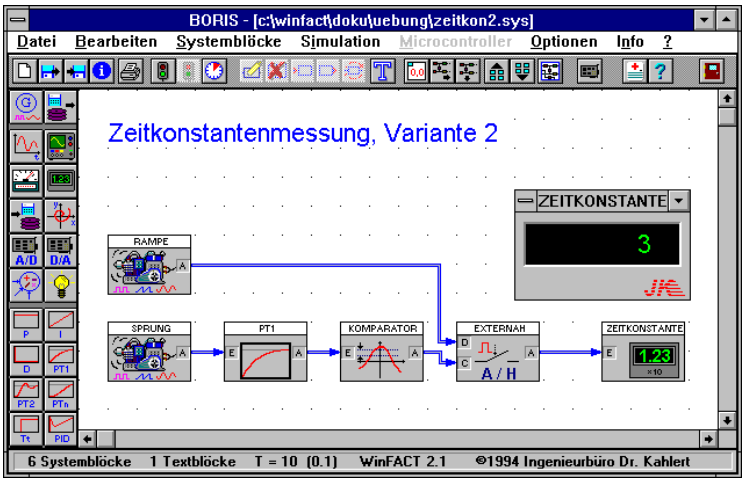
$$G(s) = \frac{K}{1+Ts}$$

anhand der Sprungantwort.

Lösungsskizze: Es existiert eine Vielzahl von Lösungsmöglichkeiten. Eine Variante besteht darin, die Sprungantwort zu differenzieren und davon das Maximum, d. h. die Anfangssteigung der Sprungantwort, zu bestimmen. Dieser Wert entspricht K/T. Damit die Schaltung für beliebiges K funktioniert, muß die Verstärkung ebenfalls bestimmt werden, indem das Maximum der Sprungantwort selbst erfaßt wird. Ein Funktionsblock ermittelt dann aus beiden Maxima die Zeitkonstante:



Eine andere Variante besteht darin, den Zeitpunkt zu bestimmen, an dem die Sprungantwort den Wert $K(1-1/e)$ erreicht:



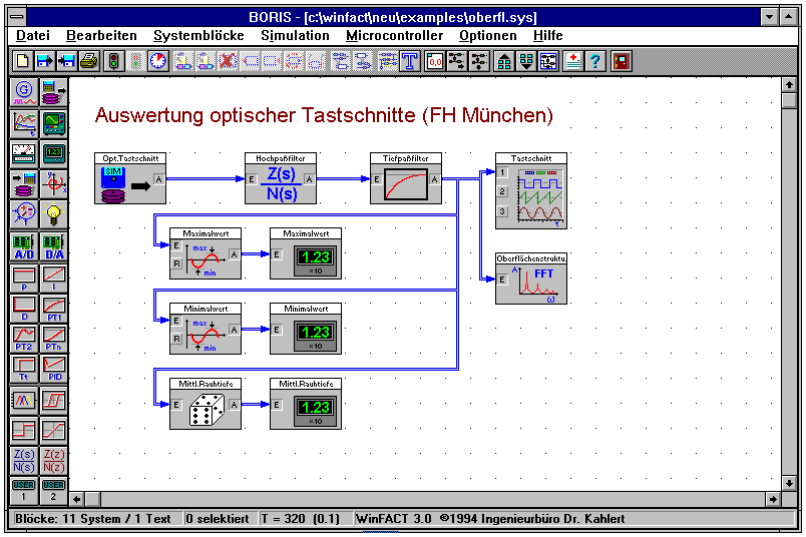
Diese Variante funktioniert jedoch nur dann, wenn die Verstärkung K des PT_1-Glieds bekannt ist, da diese im Komparator berücksichtigt werden muß.

Zugehörige Dateien: ZEITKON1.BSY
ZEITKON2.BSY

Aufgabe VI.6: Auswertung optischer Tastschnitte [11]

Aufgabenstellung: Gesucht ist eine Schaltung zur Auswertung optischer Tastschnitte, die in Dateien vom Typ SIM abgelegt sind. Es soll nach einer Bandfilterung des Tastschnitts das Amplitudenspektrum der Oberfläche visualisiert sowie minimale, maximale und mittlere Rauhtiefe ermittelt werden.

Lösungsskizze: Die Ermittlung wird mit BORIS vorgenommen. Zur Filterung wird ein Übertragungsfunktion-Block mit nachgeschaltetem PT_1-Glied benutzt. Die Auswertungsstruktur sieht wie folgt aus:

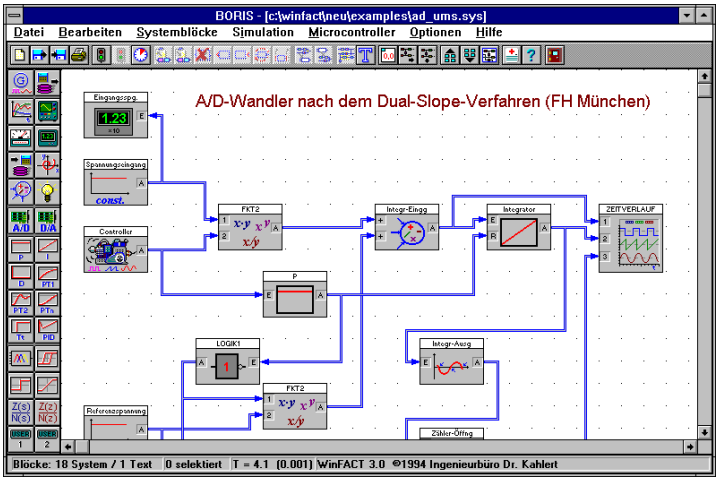


Zugehörige Dateien: OBERFL.BSY
OBERFL.SIM

Aufgabe VI.7: A/D-Wandlung nach dem Dual-Slope-Verfahren [11]

Aufgabenstellung: Erstellen Sie eine BORIS-Simulationsstruktur, die die Wirkungsweise eines A/D-Wandlers nach dem Dual-Slope-Prinzip darstellt.

Lösungsskizze: Nachfolgende Struktur zeigt einen Ausschnitt einer geeigneten Simulationsstruktur.



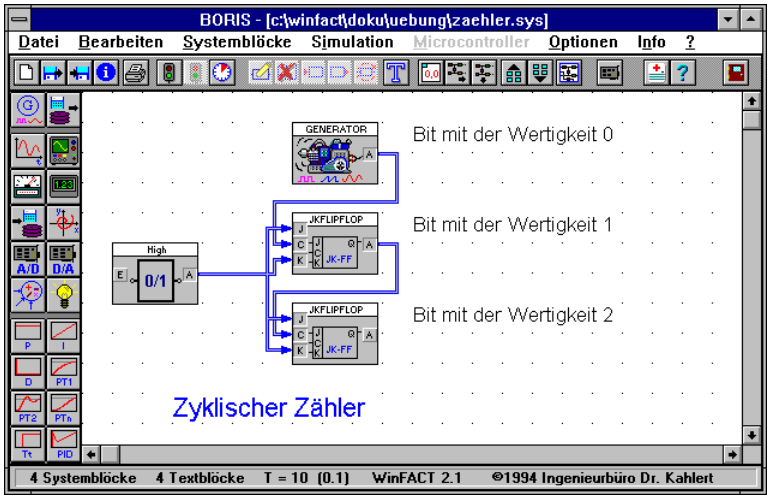
Zugehörige Dateien: AD_UMS.BSY

Kategorie VII: Digitaltechnik

Aufgabe VII.1: Zyklischer Dualzähler

Aufgaben-stellung: Ein digitales Netz mit drei Eingängen soll zum Testen alle möglichen Zustände annehmen. Entwerfen Sie mit Hilfe nur eines Generators und den Bauelementen der Digitaltechnik eine Schaltung, die diese Bedingung erfüllt.

Lösungs-skizze: Der Generator wird als Taktgeber verwendet und daher auf ein Rechtecksignal eingestellt. Sein Ausgangssignal geht auf den Takteingang eines JK-Flip-Flops, das als Frequenzteiler arbeitet (J- und K-Eingang auf 1). Dessen Ausgangssignal geht auf ein zweites JK-Flip-Flop, das ebenfalls als Frequenzteiler arbeitet.



Zugehörige Dateien: ZAEHLER.BSY

Aufgabe VII.2: 2 Bit-Addierer-Stufe

Aufgaben-stellung: Es ist ein Addierer für zwei positive 2-Bit-Binärzahlen zu erstellen. Beachten Sie dabei die möglichen Überträge. Das Ergebnis soll eine 3-Bit-Binärzahl sein. Wandeln Sie diese Schaltung anschließend so um, daß Sie das Ergebnis in eine 2-Bit-Binärzahl mit Overflow-Flag ablegen.

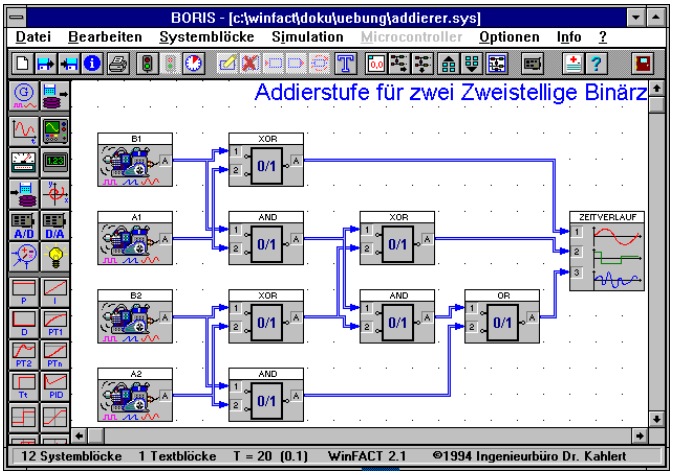
Lösungs-skizze: Die erste 2-Bit-Zahl sei $A_2 A_1$, die zweite $B_2 B_1$. Dann berechnet sich die Summe $C_2 C_1 = A_2 A_1 + B_2 B_1$ wie folgt:

$$C_1 = A_1 \neq B_1$$
$$C_2 = (A_1 \wedge B_1) \neq (A_2 \neq B_2)$$

Das Overflow-Flag berechnet sich wie folgt:

$$C_3 = (A_2 \wedge B_2) \vee \big((A_2 \neq B_2) \wedge (A_1 \wedge B_1)\big)$$

Wir kommen demnach zu folgender Schaltung:



Zugehörige Dateien: ADDIERER.BSY

Aufgabe VII.3: Codewandlung: Graycode → Dualcode

Aufgabenstellung: Eine Steueranlage, die über ein Scheibensystem einen Graycode mit drei Kohlebürsten abtastet, soll mit dem PC verbunden werden. Auf dem PC soll die momentane Stellung der Abtastung sichtbar werden. Entwerfen Sie eine entsprechende Schaltung, die die Codewandlung übernimmt.

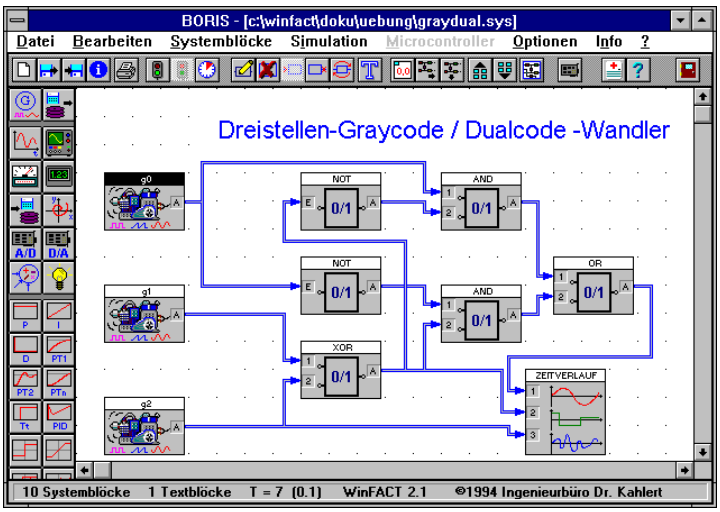
Lösungsskizze: Es sei $g_2 g_1 g_0$ die Zahl im Graycode und $d_2 d_1 d_0$ die zugehörige Dualzahl. Dann gilt folgende Umwandlungstabelle:

Gray g_2	g_1	g_0	Dual d_2	d_1	d_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	1	0	1	0
0	1	0	0	1	1
1	1	0	1	0	0
1	1	1	1	0	1
1	0	1	1	1	0
1	0	0	1	1	1

Aus dieser Tabelle lassen sich folgende Beziehungen ermitteln:

$$d_2 = g_2$$
$$d_1 = g_1 \neq g_2$$
$$d_0 = \left(g_0 \wedge \left(g_1 = g_2\right)\right) \vee \left(\overline{g_0} \wedge \left(g_1 \neq g_2\right)\right)$$

Dies entspricht folgender Systemstruktur:

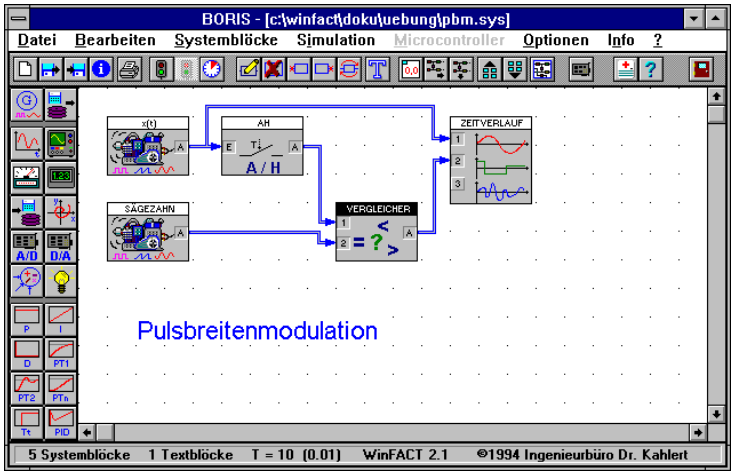


Zugehörige Dateien: GRAYDUAL.BSY

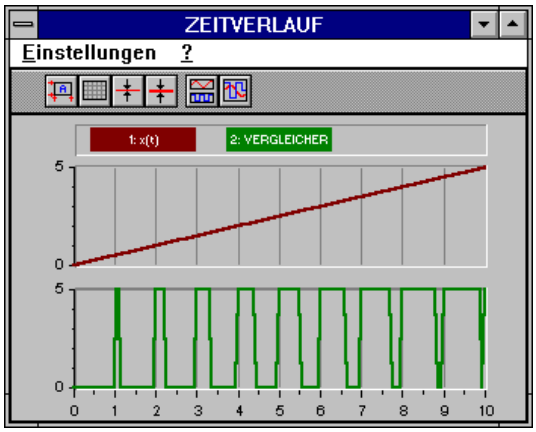
Aufgabe VII.4: Pulsbreitenmodulation

Aufgaben-stellung: Geben Sie eine Schaltung an, die aus einem analogen Eingangssignal $x(t)$ mit Amplituden zwischen 0 und 5 ein pulsbreitenmoduliertes Signal mit der Frequenz 1 Hz erzeugt.

Lösungs-skizze: Das Eingangssignal wird nach einer Abtastung mit 1 Hz auf einen Vergleicher geschickt, der es mit dem Ausgangssignal eines Sägezahlgenerators der Frequenz 1 Hz vergleicht. Je höher die Amplitude von $x(t)$ ist, umso länger bleibt der Ausgang des Vergleichers auf High-Pegel:



Die folgende Grafik zeigt das pulsbreitenmodulierte Signal bei einem linear von 0 auf 5 ansteigenden Eingangssignal $x(t)$:



Zugehörige Dateien: PBM.BSY

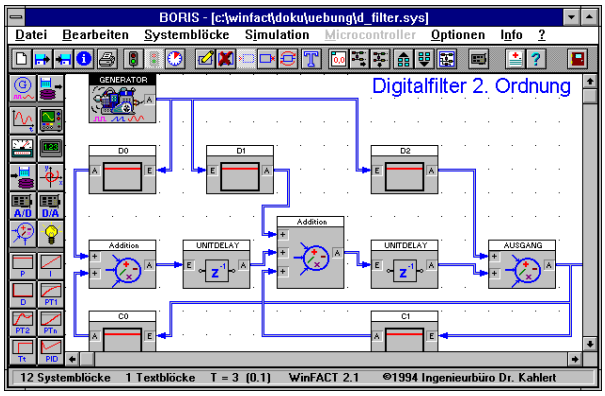
Aufgabe VII.5: Digitales Filter 2. Ordnung [13]

Aufgaben-stellung: Bestimmen Sie die Sprungantwort eines digitalen Filters mit der *z*-Übertragungsfunktion

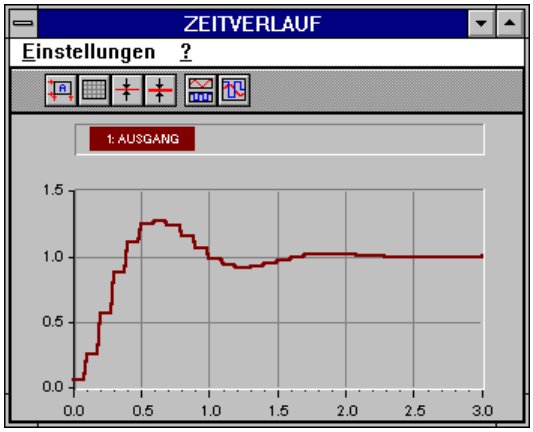
$$G(z) = \frac{0.0582 + 0.1164z^{-1} + 0.0582z^{-2}}{1 - 1.4409z^{-1} + 0.6737z^{-2}}$$

für eine Abtastzeit von 0.1. Wählen Sie dazu eine Simulationsschrittweite von 0.01 und eine Simulationsdauer von 3.

Lösungs-skizze: Das digitale Filter wird mit Hilfe von Einheitsverzögerungen z^{-1} aufgebaut:



Man erhält folgende Sprungantwort:

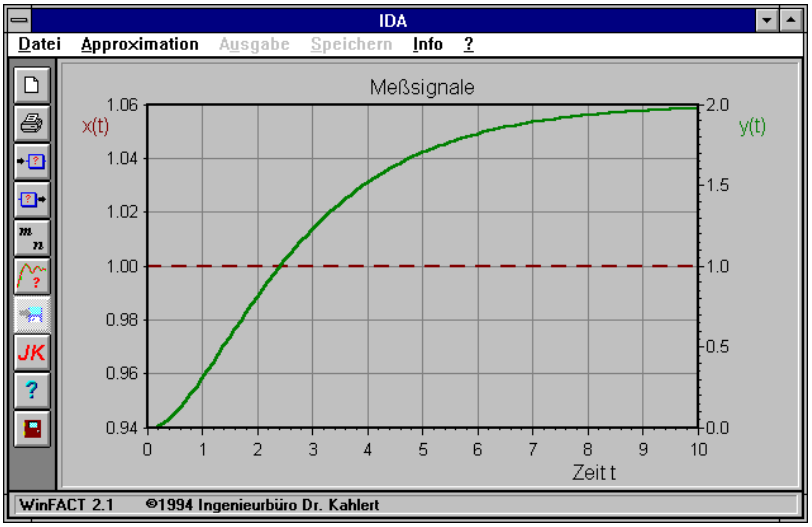


Zugehörige Dateien: D_FILTER.BSY

Kategorie VIII: Systemidentifikation

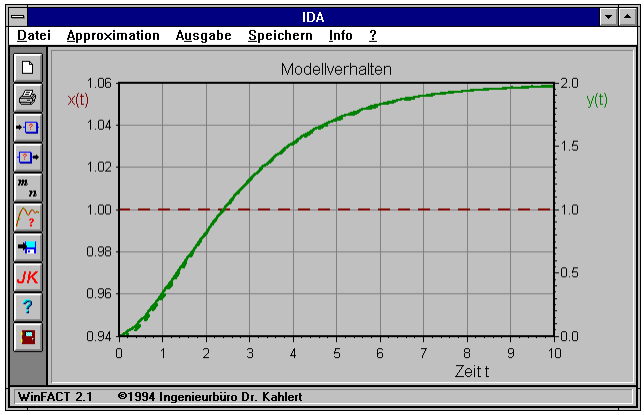
Aufgabe VIII.1: Identifikation anhand der Sprungantwort

Aufgaben-stellung: An einer Regelstrecke werde der Eingangsgrößenverlauf $x(t)$ und der Ausgangsgrößenverlauf $y(t)$ gemäß folgender Grafik gemessen:



Die Eingangsgröße ist gestrichelt, die Ausgangsgröße als Vollinie dargestellt. Ermitteln Sie die Übertragungsfunktion des zugehörigen Modells.

Lösungs-skizze: Die Identifikation läßt sich mittels IDA vornehmen. Als Zählergrad wird $m = 0$, als Nennergrad $n = 2$ gewählt. Man erhält folgendes Ergebnis:



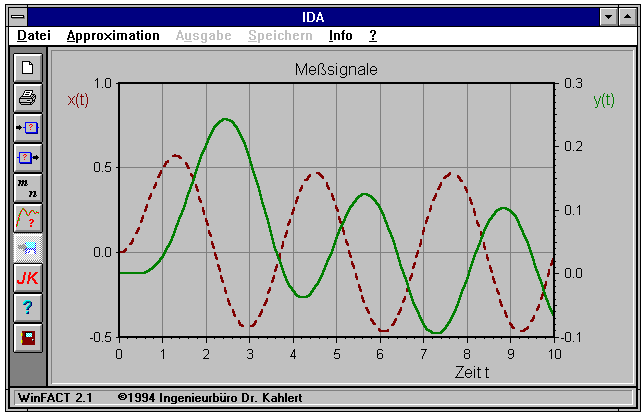
Die zugehörige Übertragungsfunktion lautet

$$G(s) = \frac{0.873}{s^2 + 0.441s + 1.282}.$$

Zugehörige Dateien: SPRUNG_X.SIM
SPRUNG_Y.SIM

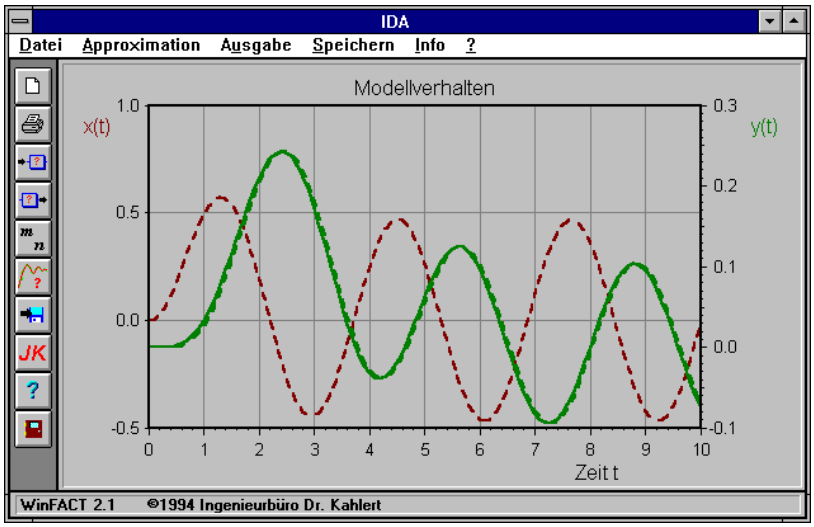
Aufgabe VIII.2: Identifikation mit harmonischen Eingangssignalen

Aufgabenstellung: An einer Regelstrecke werde der Eingangsgrößenverlauf $x(t)$ und der Ausgangsgrößenverlauf $y(t)$ gemäß folgender Grafik gemessen:



Die Eingangsgröße ist gestrichelt, die Ausgangsgröße als Vollinie dargestellt. Ermitteln Sie die Übertragungsfunktion des zugehörigen Modells.

Lösungs-skizze: Die Identifikation läßt sich mittels IDA vornehmen. Als Zählergrad wird $m = 0$, als Nennergrad $n = 3$ gewählt. Man erhält folgendes Ergebnis:



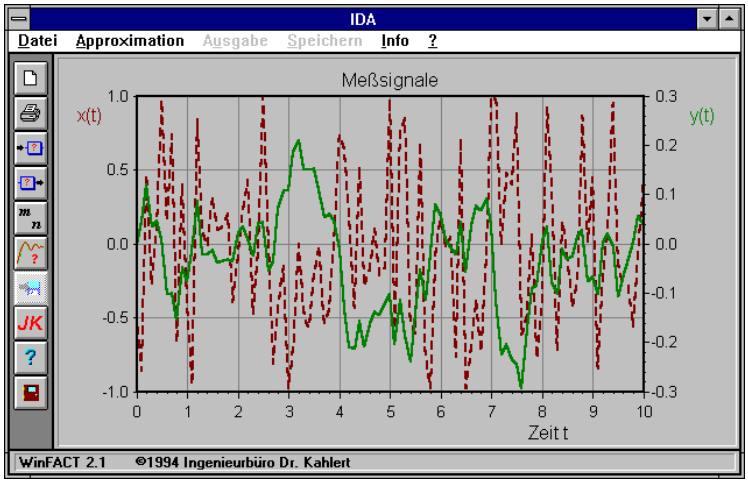
Die zugehörige Übertragungsfunktion lautet

$$G(s) = \frac{9.02}{s^3 + 9.83s^2 + 18.08s + 9.01}$$

Zugehörige Dateien: SINUS_X.SIM
SINUS_Y.SIM

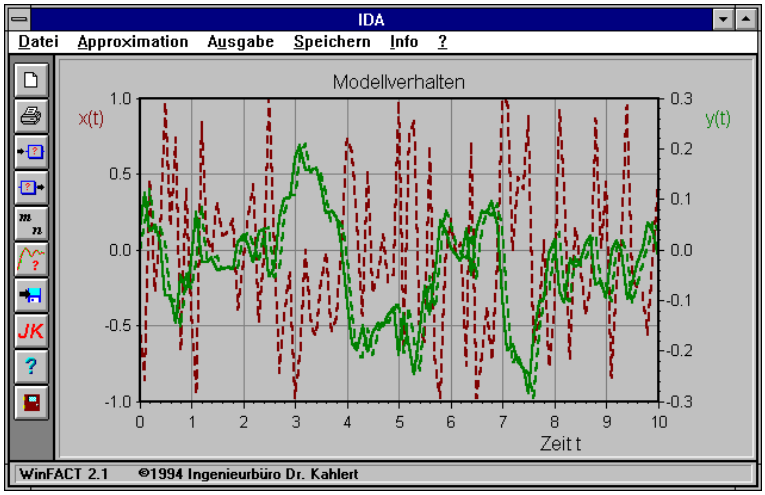
Aufgabe VIII.3: Identifikation aus verrauschten Signalen

Aufgaben-stellung: An einer Regelstrecke werde der Eingangsgrößenverlauf $x(t)$ und der Ausgangsgrößenverlauf $y(t)$ gemäß folgender Grafik gemessen:



Die Eingangsgröße ist gestrichelt, die Ausgangsgröße als Vollinie dargestellt. Ermitteln Sie die Übertragungsfunktion des zugehörigen Modells.

Lösungs-skizze: Die Identifikation läßt sich mittels IDA vornehmen. Als Zählergrad wird $m = 1$, als Nennergrad $n = 2$ gewählt. Man erhält folgendes Ergebnis:



Die zugehörige Übertragungsfunktion lautet

$$G(s) = \frac{-1.08s + 0.986}{s^2 + 1.777s + 1.025}.$$

Zugehörige Dateien: NOISE_X.SIM
NOISE_Y.SIM

Literatur- und Quellenverzeichnis

[1] N.N.: WINDOWS 95 Benutzerhandbuch

[2] Kahlert, J.: *Fuzzy Control für Ingenieure*. Vieweg Verlag Wiesbaden 1995

[3] Schmidt, G.: *Simulationstechnik*. Oldenbourg Verlag 1980

[4] Driankov, D. et al.: *An Introduction to Fuzzy Control*. Springer Verlag 1994

[5] Golubev, B.; Horowitz, I.: *Plant rational transfer approximation from input-output data*. International Journal on Control 4/1982.

[6] Föllinger, O.: *Regelungstechnik*. Hüthig Buchverlag Heidelberg 1990

[7] Unbehauen, H.: *Regelungstechnik*. Vieweg-Verlag Wiesbaden 1992

[8] Preuß, H. P.: *Fuzzy Control - heuristische Regelung mittels unscharfer Logik*. Automatisierungstechnische Praxis atp 4-5/1992

[9] Zimmermann, H. J.: *Fuzzy-Technologien*. VDE-Verlag 1993

[10] Bossel, H.: *Modellbildung und Simulation*. Vieweg-Verlag 1993

[11] FH München, Labor für Robotik (Prof. Dr. O. Schwab)

[12] Kahlert, J.: Skriptum zur Vorlesung "Simulationstechnik", Universität Dortmund 1993

[13] Tietze, U.; Schenk, Ch.: *Halbleiter-Schaltungstechnik*. 5. Auflage, Springer-Verlag Berlin Heidelberg New York

[14] Kahlert, J.: *Globale vektorielle Optimierung mit Evolutionsstrategien*. Automatisierungstechnik 3/95

Index

E

F

M

N

O

P

R

S

T

U

V

W

X

Z

