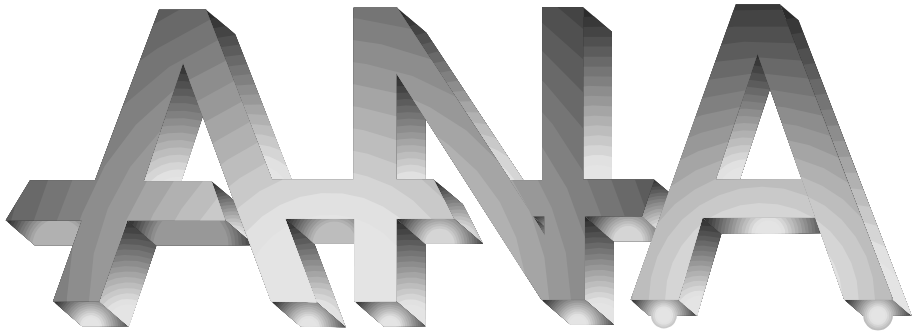


Version 2.12

Benutzerhandbuch

Zweite, überarbeitete Auflage

Johann M. Marinits
Johannes W. Goldynia

TU Wien, Institut für
Elektrische Regelungstechnik

ANA 2.12 Benutzerhandbuch (V 2.1)

2. *Auflage*

Inhaltsverzeichnis

1 Urheberrechts-Vermerk

1.1 Urheberrecht

Das Urheberrecht von ANA liegt bei:

Institut für Elektrische Regelungstechnik
Technische Universität Wien
Gußhausstr. 27-29/375
A-1040 Wien

Autoren: Johannes W. Goldynia und Johann M. Marinits
email: goldynia@iert.tuwien.ac.at, marinits@iert.tuwien.ac.at

Dies bedeutet, daß es Ihnen verboten ist Veränderungen jeglicher Art, die das Programm oder die Dokumentation betreffen, vorzunehmen. Im Besonderen ist das Entfernen der Dokumentation sowie des Urheberrechts-Vermerks untersagt. Weder das Programm noch Teile davon dürfen rückübersetzt werden.

1.2 Weitergabe

Die aktuelle Version des Programmpakets erhalten Sie über die Internetadresse:

`ftp://ftp.iert.tuwien.ac.at/ana2`

Information und Software finden sich auch unter:

`http://www.iert.tuwien.ac.at/ana2`

Dieses Programmpaket darf kostenfrei weitergegeben werden. Das bedeutet, daß Sie dieses Programmpaket ausschließlich unter Beachtung folgender Auflagen weitergeben dürfen:

1. Jede Weitergabe muß alle im Programmpaket enthaltenen Dateien in unveränderter Form beinhalten, einschließlich des Urheberrechts-Hinweises. Sie dürfen dem Programmpaket keine Dateien hinzufügen.
2. Das Programmpaket darf frei über Mailboxen, dem Internet/UseNet oder ähnlichen elektronischen Datennetzen weitergegeben werden. Zeitschriften mit beigelegten Datenträgern dürfen das Programmpaket ohne schriftliche Genehmigung der Autoren nicht weitergeben. Das gilt auch für alle Datendienste, die für die Datenübetragung von Dateien Gebühren einheben.
3. Für die Weitergabe des Programmpakets dürfen Sie keinerlei Gebühren verlangen.

1.3 Haftungsausschluß

Durch Verwendung dieses Produkts übernehmen Sie die volle Haftung für jeden Schaden oder Verlust, der durch die Verwendung bzw. Nicht-Verwendbarkeit entsteht. Die Autoren des Programmpakets können dafür nicht verantwortlich gemacht werden.

Tutorial

2 Ein erstes Beispiel

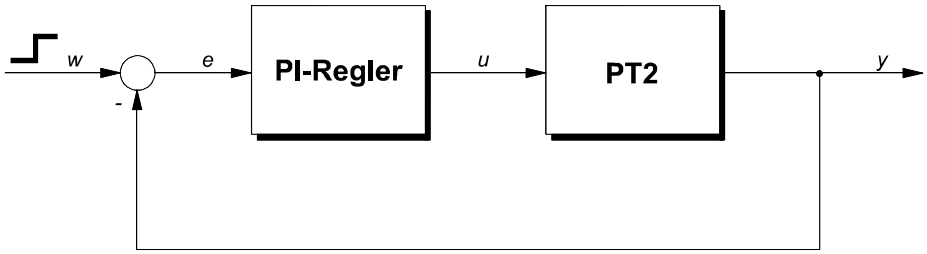


Abbildung 2.1: Ein erstes Beispiel

Für eine nicht schwingungsfähige PT_2 Strecke soll die Sprungantwort für einen betragsoptimalen Regler ermittelt werden.

Für die Regelstrecke $G(s) = \frac{1}{(1+2s)(1+0.1s)}$ lautet der betragsoptimale Regler $K(s) = 10(1+1/2s)$.

2.1 Die Eingabe des Modells

ANA verwendet zur Darstellung eines dynamischen Systems das Blockschaltbild. Ein Modell wird eingegeben, indem man einzelne Blöcke aus diversen Blockbibliotheken auf der Arbeitsfläche anordnet und verbindet.

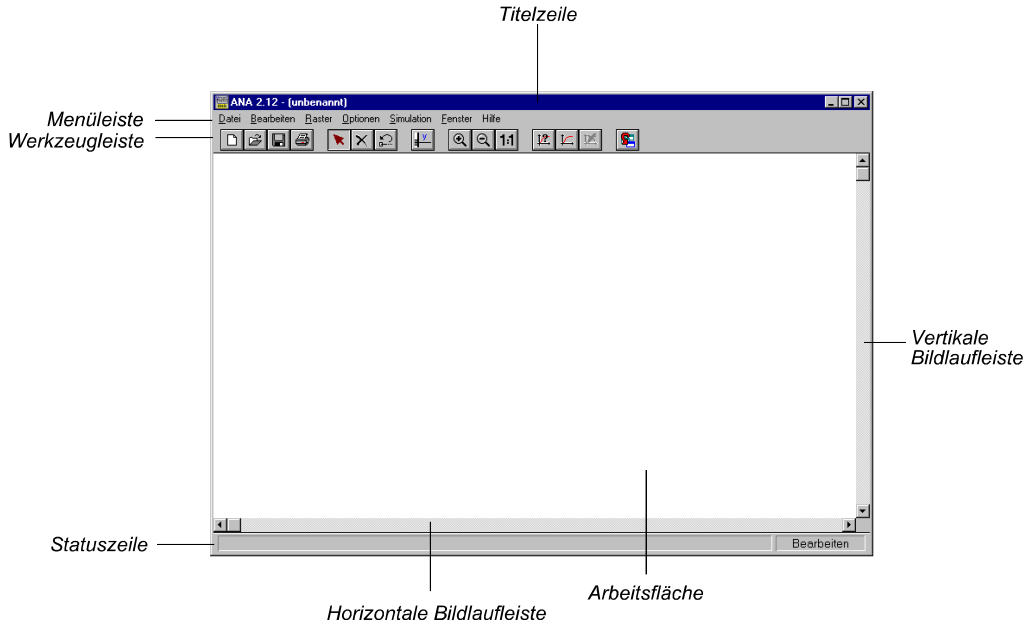


Abbildung 2.2: Bildschirmaufbau

Startet man ANA, so öffnet sich ein wie in Abbildung 2.1 angegebenes Fenster. Befindet sich der Mauszeiger über der Arbeitsfläche und drückt man die rechte Maustaste, so erhält man ein lokales Menü. Durch Auswahl des Menüpunktes „**Blockbibliotheken ...“** erscheint der Blockauswahldialog (Abbildung 2.3).

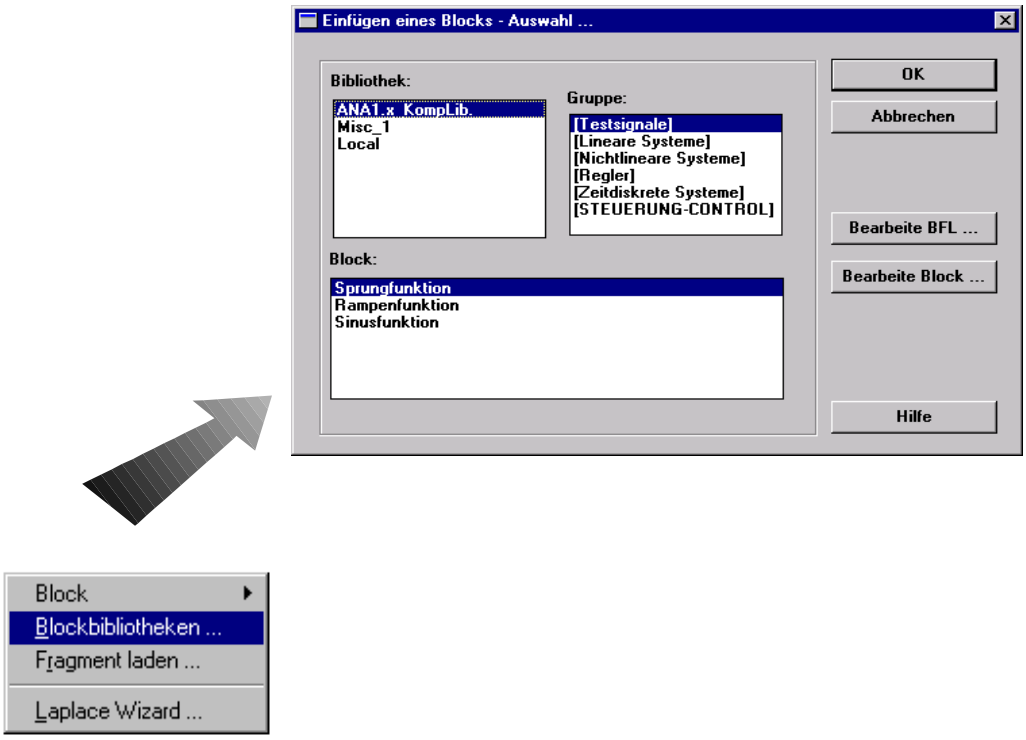


Abbildung 2.3: Lokales Arbeitsflächenmenü und Blockauswahldialog

Wählt man aus der Bibliothek ANA1.x_KompLib, Gruppe Testsignale, den Block *Sprungfunktion*, so erscheint dieser nach Drücken des OK-Buttons auf der Arbeitsfläche.

Das Symbol ist in drei Bereiche eingeteilt (Abbildung 2.5):

- Die Eingangskonnektoren (symbolisiert durch kleine Dreiecke) auf der linken Seite,
- die Ausgangskonnektoren (symbolisiert durch kleine Rechtecke) auf der rechten Seite,
- das Beschriftungsfeld in der Mitte.

Bewegt man den Mauszeiger über den Block, so ändert dieser in Abhängigkeit der jeweils verfügbaren Funktionen sein Aussehen. Eine Zusammenfassung der daraus resultierenden Möglichkeiten ist im Referenzteil angegeben.

Zum Einstellen der Parameter drückt man die rechte Maustaste während sich der Mauszeiger über dem Beschriftungsfeld des Blocks befindet. Es erscheint ein lokales Menü und durch Auswählen des Menüpunktes „**Parameter ...“** (linke Maustaste) öffnet sich der Parameterdialog (Abbildung 2.4). Für dieses Beispiel wird die Einstellung A = 1, T = 0 und A0 = 0 verwendet.

Als nächster Block wird ein Addierer mit 2 Eingängen gewählt und die Parameter auf k1 = 1 und k2 = -1 gesetzt. Man plaziert nun den Addierer Block rechts vom Sprungblock. Um einen Block zu verschieben, drückt man die linke Maustaste

während sich der Mauszeiger über dem Beschriftungsfeld des Blockes befindet und hält die Taste gedrückt. Der Block bewegt sich mit dem Mauszeiger mit. Befindet sich der Block an der gewünschten Position, läßt man die Maustaste wieder los.

Will man mehrere Blöcke gleichzeitig verschieben, so müssen diese zuvor markiert werden. Dazu drückt man die linke Maustaste während sich der Mauszeiger über der leeren Arbeitsfläche befindet und hält die Taste gedrückt. Bewegt man nun den Mauszeiger, so wird ein strichliertes Rechteck angezeigt, das den *Fangbereich* darstellt. Läßt man nun die linke Maustaste los, so werden alle Blöcke, die sich vor dem Loslassen der linken Maustaste im Fangbereich befunden haben, markiert.

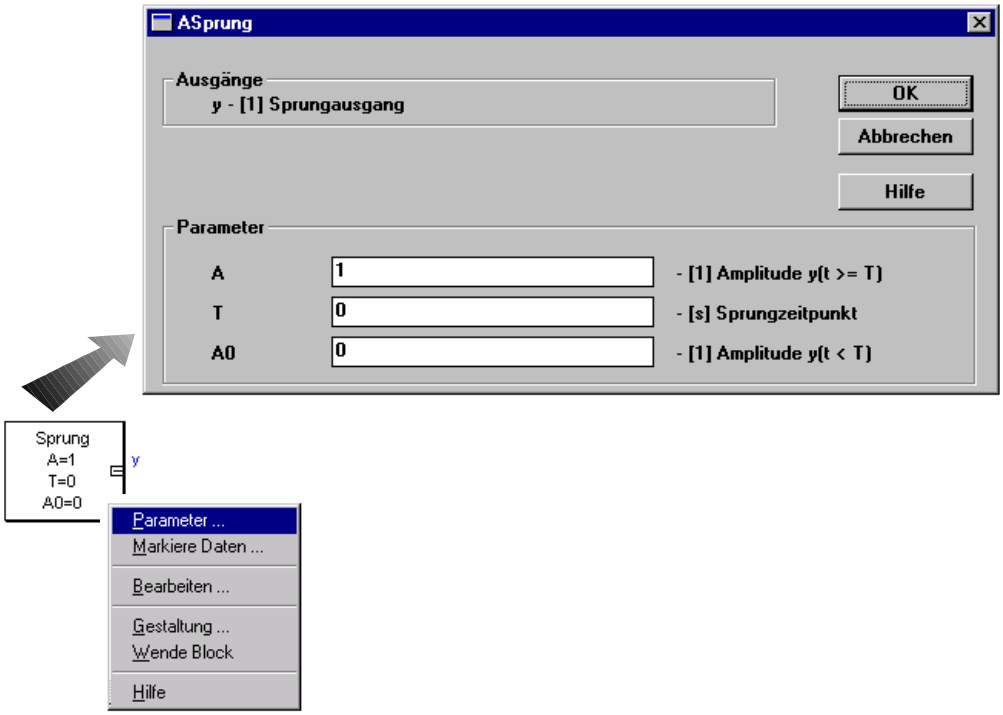


Abbildung 2.4: Lokales Blockmenü und Parameterdialog

Nun wird der Ausgang des Sprungblocks mit dem ersten Eingang des Addiererblocks verbunden:

1. Man bewegt den Mauszeiger über den Ausgangskonnektor des Sprungblocks (der Mauszeiger verwandelt sich in einen Stift) und drückt die linke Maustaste – der Konnektor wird rot umrandet und es wird eine graue Verbindungslinie vom Ausgangskonnektor zur Spitze des Mauszeigers gezeichnet.
2. Man bewegt den Mauszeiger über den Eingangskonnektor des Addiererblocks und drückt die linke Maustaste – zwischen den beiden Konnektoren wird eine Verbindungslinie gezeichnet.

Auf analoge Weise werden die restlichen Blöcke eingegeben (für das PT_2-Element wurden aufgrund der Darstellung in der Angabe vorteilhaft zwei PT_1-Blöcke, für den PI-Regler wurde ein PDT_1-Regler mit DT_1-Anteil $T_v = 0$ verwendet).

Für jedes Modell muß ein Steuerungsblock definiert werden. Er wird aus der Blockbibliothek ANA1.x_KompLib\Steuerung\Simul.-Steuerung geladen. Dadurch werden Simulationsdauer und Kommunikationsintervall als Parameter festgelegt.

Wie man in Abbildung 2.5 sieht, ist es auch möglich Verbindungslinien mit „Ecken“ zu legen. Dazu klickt man – während man eine Verbindungslinie verlegt – mit der linken Maustaste an jene Positionen, an denen man einen Stützpunkt wünscht. Man kann auch nachträglich einen Stützpunkt einfügen, indem man den Mauszeiger an jene Stelle bewegt, an die man ihn einfügen möchte (Mauszeiger wechselt auf Stützpunkt-Einfügen-Symbol) und die linke Maustaste drückt. Ein so eingefügter Punkt läßt sich analog zu einem Block verschieben (Abbildung 2.6).

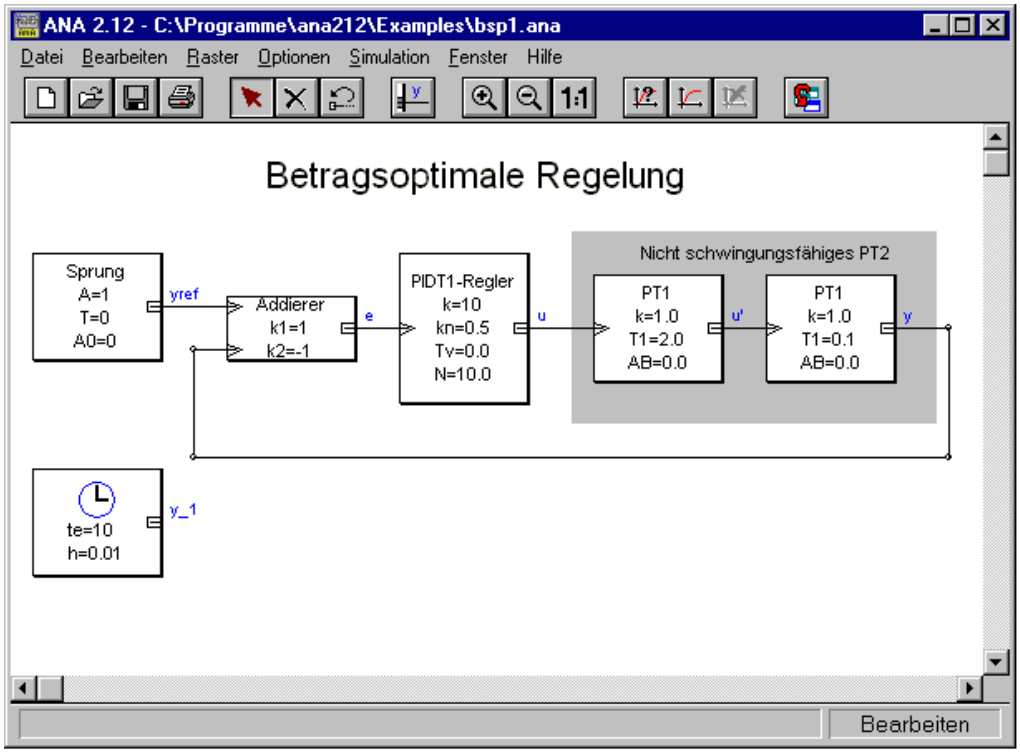


Abbildung 2.5: Das fertig eingegebene Modell

Möchte man die Verbindungslinie, die man gerade zeichnet löschen, so klickt man einfach auf jenen Konnektor, bei dem man mit dieser Verbindungslinie begonnen hat.

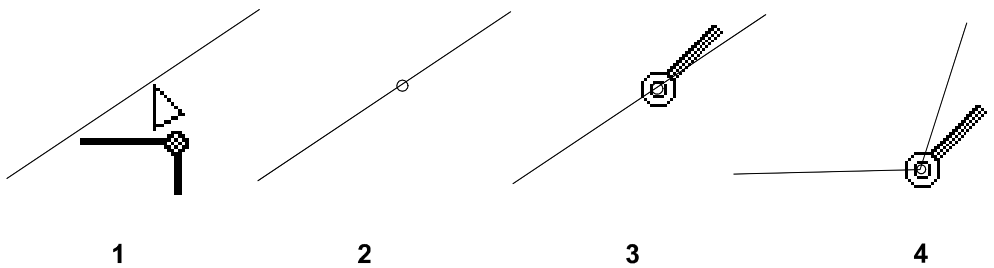


Abbildung 2.6: Stützpunkt einfügen und verschieben

Die Beschriftung „Betragsoptimale Regelung“ erhält man über einen Schriftfeldblock aus der Bibliothek Misc_1/Dokumentation. Der anzuzeigende Text ist Parameter dieses Blockes. Die farbige Unterlegung der Strecke erreicht man über einen Rahmenblock aus derselben Bibliothek.

ANA kennt drei Betriebsmodi:

- *Edit-Modus:* Blöcke können plaziert, parametriert und verbunden werden. Dieser Modus ist beim Starten von ANA eingestellt.
- *Lösch-Modus:* Blöcke und Verbindungen können gelöscht werden.
- *Switch-Conn-Modus:* Die Verbindungen zweier Konnektoren können vertauscht werden.

Um ein eventuell falsch eingegebenes Objekt zu löschen, wählt man den Menüpunkt „**Bearbeiten → Lösche Objekte - Werkzeug**“ oder betätigt den entsprechenden Button der Werkzeugleiste (siehe Referenz) oder drückt einfach die Taste **F2**. Man befindet sich jetzt im Modus *Löschen* und einzelne Objekte können durch Drücken der linken Maustaste entfernt werden.

ANA ordnet jedem Ausgang eines Blockes automatisch einen Signalnamen in der Form *Sig n* zu, im eben eingegebenen Beispiel *Sig 1* bis *Sig 6*. Um den Ausgängen andere Signalnamen zuzuweisen, wählt man „**Bearbeiten → Signalnamen bearbeiten ...**“. Im darauffolgenden Dialog kann man die Signalnamen editieren.

Zum Abspeichern des Beispiels wählt man „**Datei → Speichere Schaltung ...**“ – es erscheint ein Standard Dateidialog mit dessen Hilfe man das erstellte Modell unter dem Namen `bsp1.ana` abspeichert.

2.2 Simulation des Zeitverhaltens

Die Simulation zeichnet nur Signale (bzw. Variablen) auf, die vorher markiert wurden. Um ein Signal zu markieren, bewegt man den Mauszeiger über den entsprechenden Ausgangskonnektor und drückt die rechte Maustaste. Ein so selektierter Ausgangskonnektor wird durch ein grau gefülltes Rechteck symbolisiert. Alternativ dazu ist es auch möglich, die rechte Maustaste zu drücken, während sich der Mauszeiger über dem Beschriftungsfeld des Blocks befindet. Es erscheint ein lokales Menü und durch Auswählen des Menüpunktes „**Markiere Daten ...**“ öffnet sich der Markierungsdialog, der nicht nur Zugriff auf die Signale, sondern auch auf blockinterne Variablen und Parameter ermöglicht. Markierte Signale sind durch einen Stern (*) gekennzeichnet (Abbildung 2.7). In diesem Beispiel sollen Regelgröße und Sollwert aufgezeichnet werden.

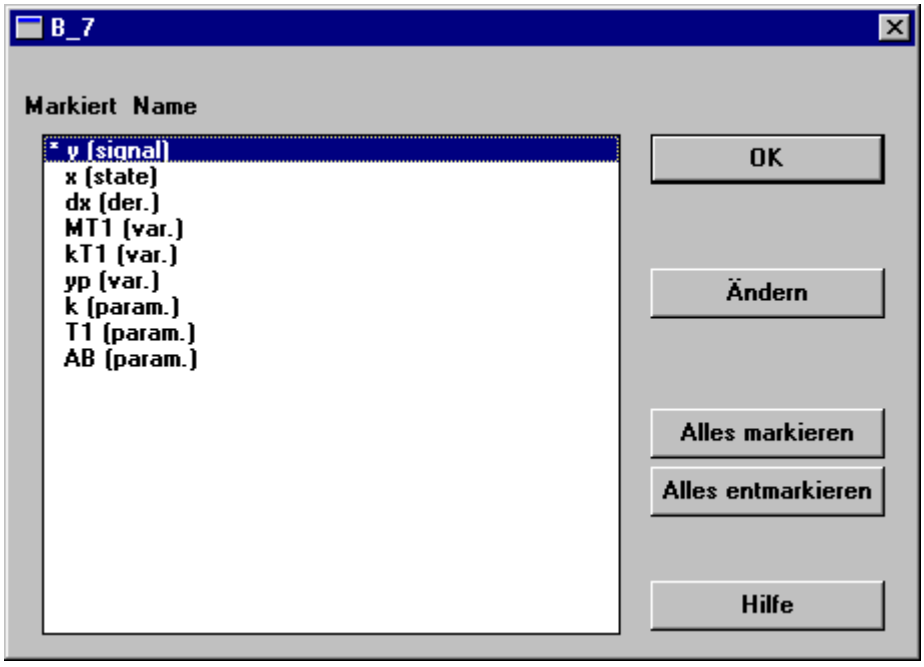


Abbildung 2.7: Markierungsdialog

Durch Auswählen des Menüpunktes „**Simulation → Start ...**“ bzw. durch Drücken des entsprechenden Buttons in der Werkzeugleiste wird das Modell übersetzt und falls kein Fehler aufgetreten ist, öffnet sich der Simulationsdialog. Dieser Dialog ermöglicht die Einstellung diverser Simulationsparameter (siehe Referenz). Im untersten Feld (*ANAOszi Signal für x-Achse*) stellt man jene Größe ein, die auf der *x*-Achse aufgetragen werden soll. Durch die Standardeinstellung *time* werden die beiden markierten Signale in Abhängigkeit der Zeit dargestellt.

Nach Bestätigen mit **OK** wird die eigentliche Simulation gestartet. Es öffnen sich das ANAOszi Fenster, indem die Aufzeichnung der markierten Signale mitverfolgt werden kann. Durch Auswählen des Menüpunktes „**Simulation → Stop**“ im ANA Fenster kann die Simulation frühzeitig abgebrochen werden.

Nach Beendigung der Simulation werden die im ANAOszi dargestellten Signale automatisch skaliert (Default-Einstellung).

Jedem Signal wird automatisch eine bestimmte Farbe zugewiesen. Diese, sowie der Maßstab für *x*- und *y*-Achse, kann durch Auswählen von „**Optionen → Parameter**“ verändert werden.

Durch Auswahl des Menüpunktes „**Datei → Drucken**“ kann der ANAOszi-Schirm gedruckt werden. Vorher wird ein Dialog zum Einstellen des Druckertreibers (Windows Standarddialog) angezeigt. Das Simulationsergebnis zeigt Abbildung 2.8.

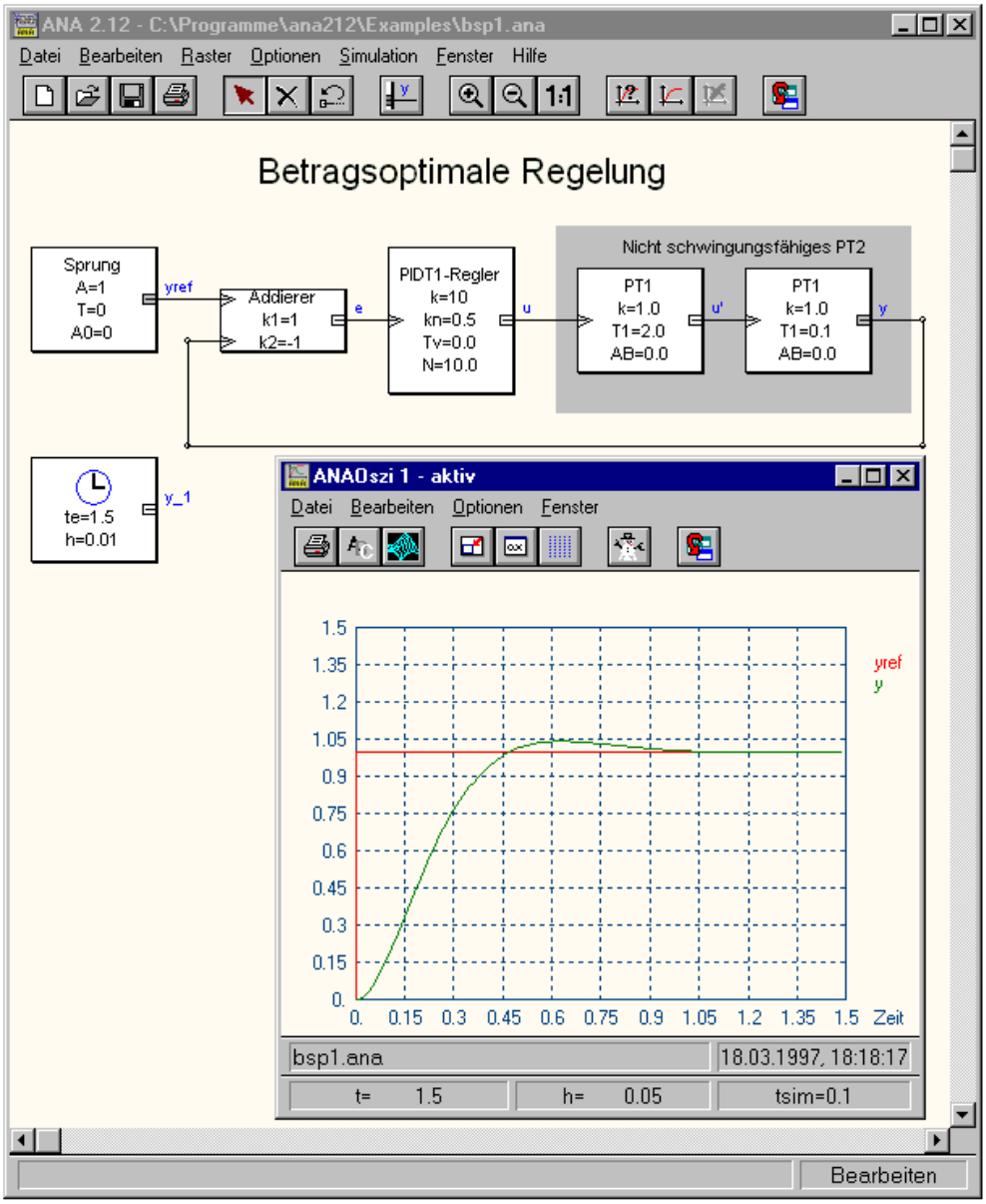


Abbildung 2.8: Simulationsergebnis

Referenz

3 Grafische Bedienoberfläche

Die grafische Bedienoberfläche besteht im wesentlichen aus folgenden Komponenten (siehe Abbildung 3.1):

- ANAIde – Die integrierte Entwicklungsumgebung dient zur Eingabe des Modells (Blockschaltbildeditor) und zur Steuerung des Simulators (Integrationsverfahren auswählen, Simulation starten/beenden, ...).
- ANAOszi – Dient zur Darstellung der Simulationsdaten (Signale).
- ANAWizards – Diese „Plug In"-Module dienen zur Erweiterung der Funktionalität von ANA und übernehmen spezielle Aufgaben. In der aktuellen Version von ANA ist ein solcher Zauberer im Einsatz, der LaplaceWizard.

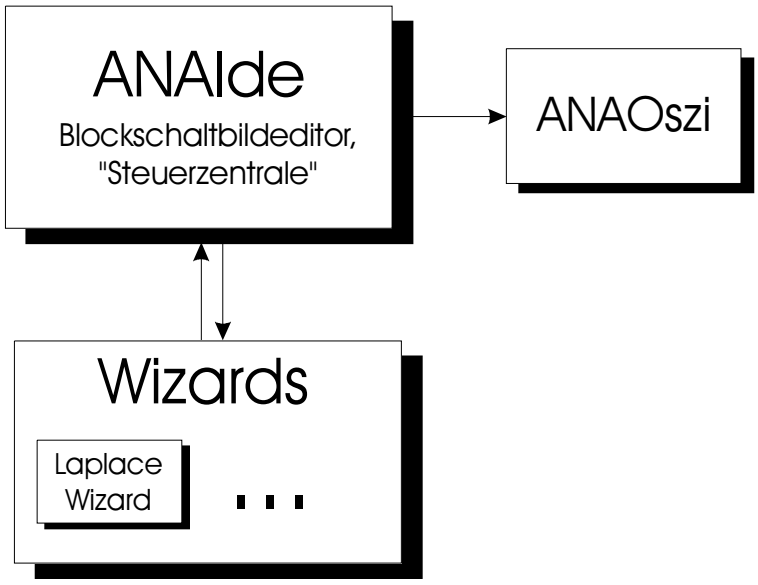


Abbildung 3.1: Aufbau der grafischen Oberfläche von ANA.

3.1 ANAIde

3.1.1 Menüs

Den Zusammenhang zwischen den einzelnen Menüpunkten und den Buttons in der Werkzeugleiste zeigt Abbildung 3.2.

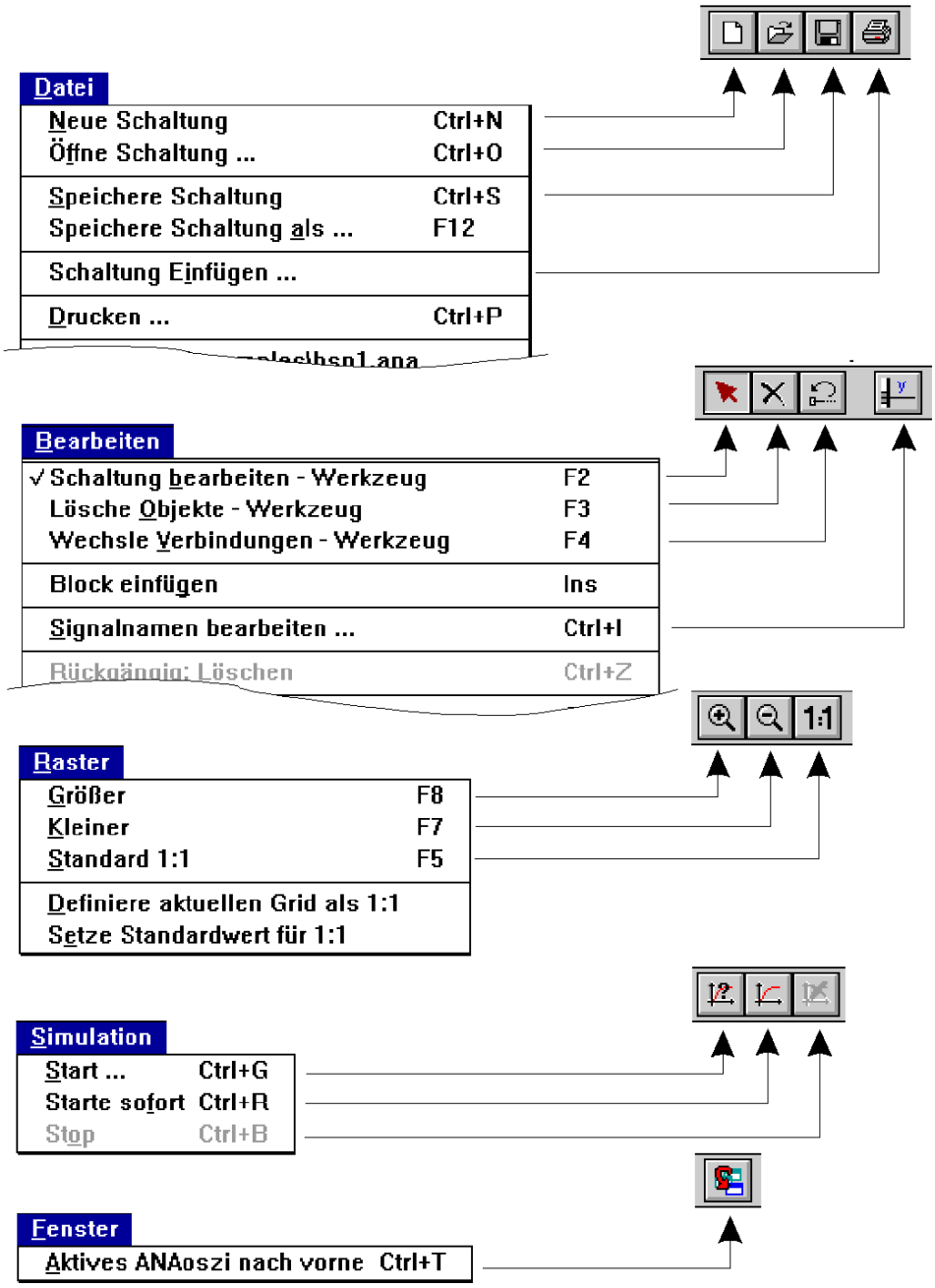


Abbildung 3.2: ANAIde-Menüs und dazugehörige Buttons der Werkzeugleiste

3.1.1.1 Datei Menü

Neue Schaltung: Das aktuelle Modell wird gelöscht (Sicherheitsabfrage bei noch nicht gespeichertem Modell) und der Dateiname des neuen Modells (in der Titelzeile des Fensters angeführt) wird auf „`(unbenannt)`“ gesetzt.

Öffne Schaltung: Über einen Dateidialog läßt sich ein Modell laden. ANA-Modelle haben die Dateierweiterung `*.ana`. Wird ein ANA-1.x-Modell geladen, so wird es automatisch in das neue Modellformat konvertiert.

Speichere Schaltung: Das aktuelle Modell wird unter seinem aktuellen Namen gespeichert. Wird das Modell zum ersten Mal gespeichert, öffnet sich ein Dateidialog.
Hinweis: Wurde ein ANA-1.x-Modell geladen, so wird es im neuen Dateiformat gespeichert.

Speichere Schaltung als: Über einen Dateidialog kann das aktuelle Modell gespeichert werden.

Schaltung einfügen: Über einen Dateidialog läßt sich ein Modell laden, das in das aktuelle Modell eingefügt wird. Treten Konflikte bei den Signalnamen auf, so werden diese automatisch aufgelöst.

Drucken: Das aktuelle Modell wird ausgedruckt. Die Größe des Ausdrucks richtet sich nach der gewählten Vergrößerungsstufe.

Beenden: ANA wird verlassen.

3.1.1.2 Bearbeiten Menü

Schaltung bearbeiten – Werkzeug: Schaltet in den Modus „Editieren“. Den Zusammenhang zwischen dem jeweiligen Modus mit den Cursorsymbolen und der Funktion der Maustasten zeigt Abbildung 3.3.

Lösche Objekte – Werkzeug: Schaltet in den Modus „Löschen“.

Wechsle Verbindungen – Werkzeug: Schaltet in den Modus „Verbindungslinien wechseln“.

Block einfügen: Ruft das lokale Arbeitsflächenmenü (siehe Abschnitt 3.1.1.7) auf.

Signalnamen bearbeiten: Über einen Dialog lassen sich die Namen der einzelnen Signale editieren (Abschnitt 3.1.4.5).

Rückgängig: Löschen: Stellt gelöschte Objekte wieder her. Wurden mehrere Objekte auf einmal gelöscht (z.B. mehrere markierte Blöcke oder Verbindungslinien), so können diese über „**Rückgängig: Löschen**“ *einzeln* wiederhergestellt werden.

Ausschneiden : Markierte Objekte werden gelöscht *und* in das ANAClipboard kopiert.

Kopieren: Markierte Objekte werden in das ANAClipboard kopiert.

Einfügen: Objekte, die im ANAClipboard abgelegt sind, werden auf der Arbeitsfläche eingefügt.

Löschen: Markierte Objekte werden gelöscht, im Gegensatz zu „**Ausschneiden“** allerdings *nicht* in das ANAClipboard kopiert.

Duplizieren: Markierte Objekte werden dupliziert (dabei werden Sie *nicht* in das ANAClipboard kopiert).

Fragment speichern als: Markierte Teile einer Schaltung können als Fragment (Teilschaltung) abgespeichert werden. Fragmentdateien haben die Dateierweiterung `*.afr`.

In die Zwischenablage kopieren: Markierte Objekte werden (als Bitmap Grafik) in die System-Zwischenablage kopiert und können damit in andere Anwendungen, beispielsweise in eine Textverarbeitung, importiert werden.

Standardabmessungen für markierte Blöcke: Markierte Blöcke werden auf Ihre Standardabmessung zurückgesetzt.

3.1.1.3 Raster Menü

Größer: Es wird der Rasterabstand (und somit die dargestellten Objekte) vergrößert.

Kleiner: Der Rasterabstand wird verkleinert.

Standard 1:1: Der Standardrasterabstand wird gewählt.

Definiere aktuellen Grid als 1:1: Die aktuelle Rastereinstellung wird als „Standard 1:1“ definiert. Dadurch ist es möglich, die Objektgrößen optimal in Abhängigkeit der vorhandenen Grafikkartenauflösung einzustellen.

Setzte Standardwert für 1:1: Die Rastereinstellung für „Standard 1:1“ wird auf den voreingestellten Wert gesetzt.

3.1.1.4 Optionen Menü

Raster nicht aktiv: Durch Auswahl dieses Menüpunktes werden Objekte beim „Ziehen“ mit der Maus „glatt“ bewegt und springen nicht entlang der Rasterlinien. Diese Auswahl hat aber keinen Einfluß auf die endgültige Position des Objektes; dieses wird immer am Raster ausgerichtet.

Verstecke Verbindungslinien: Durch Auswahl dieses Menüpunktes werden alle Verbindungslinien unsichtbar gezeichnet. Gleichzeitig wird „**Zeige Signalnamen an Eingängen“** gewählt. Diese Betriebsart entspricht jener von ANA-1.x (wo es noch keine Verbindungslinien gab) und kann beim Bearbeiten alter ANA Beispiele nützlich sein.

Zeige Signalnamen an Eingängen: Ist dieser Menüpunkt ausgewählt, so werden zusätzlich die Eingangskonnektoren mit Signalnamen beschriftet.

Hinweis: Beim Laden von ANA-1.x Beispielen werden die Menüpunkte „**Verstecke Verbindungslinien“** und „**Zeige Signalnamen an Eingängen“** automatisch ausgewählt.

3.1.1.5 Simulation Menü

Start: Es wird der Simulationsdialog (Abschnitt 3.1.4.6) aufgerufen.

Starte sofort: Die Simulation wird mit den aktuellen Einstellungen des Simulationsdialogs sofort gestartet.

Stop: Die laufende Simulation wird abgebrochen.

3.1.1.6 Fenster Menü

Aktives ANAOszi nach vorne: Das aktive ANAOszi Fenster wird in den Vordergrund geholt. In diesem Menü werden außerdem alle ANAOszis aufgelistet und können durch Auswahl nach vorne geholt werden.

3.1.1.7 Lokales Arbeitsflächenmenü

Block: Als Untermenüpunkte sind die verfügbaren Blockbibliotheken mit ihren Gruppen und Blöcken aufgelistet. Möchte man etwa einen Sprungfunktionsblock auf der Arbeitsfläche anordnen, so wählt man:

„**Block→ANA1.x_KompLib→Testsignale→Sprungfunktion**“.

Diese Menüstruktur entspricht dem Aufbau des Blockauswahldialogs und stellt somit eine Alternative zu diesem dar.

Blockbibliotheken: Es wird der Blockauswahldialog aufgerufen.

Fragment laden: Teilschaltungen, die als Fragment (Dateierweiterung `*.afr`) abgespeichert wurden, werden geladen.

Laplace Wizard: Es wird der LaplaceWizard-Dialog aufgerufen.

3.1.1.8 Lokales Blockmenü

Drückt man die rechte Maustaste während sich der Mauszeiger über dem Beschriftungsfeld eines Blocks befindet, so erscheint ein lokales Menü (Abbildung 2.4: Lokales Blockmenü und Parameterdialog) mit folgenden Einträgen:

Parameter: Es wird der Parameterdialog aufgerufen (Abschnitt 3.1.4.2).

Markiere Daten: Es wird der Markierungsdialog aufgerufen (Abschnitt 3.1.4.3).

Bearbeiten: Es wird der entsprechende Wizard-Dialog aufgerufen, falls es sich um einen Wizard-Block handelt. Ansonsten erhält man eine Fehlermeldung.

Gestaltung: Es wird der Blockgestaltungsdialog aufgerufen (Abschnitt 3.1.4.4).

Wende Block: Standardmäßig befinden sich die Eingangskonnektoren auf der linken Seite eines Blockes und die Ausgangskonnektoren auf der rechten Seite. Durch Auswählen von „Wende Block“ wird die Anordnung der Ein- und Ausgangskonnektoren vertauscht.

Hilfe: Es wird Information zum Block angezeigt.

3.1.2 Cursorsymbole

Die Cursorsymbole und den Zusammenhang mit dem jeweiligen Betriebsmodus und den Maustasten zeigt Abbildung 3.3.

3.1.3 Markieren und Entmarkieren

Um Objekte zu markieren drückt man die linke Maustaste während sich der Mauszeiger über der leeren Arbeitsfläche befindet und hält die Taste gedrückt. Bewegt man nun den Mauszeiger, so wird ein strichliertes Rechteck angezeigt, das den *Fangbereich* darstellt. Läßt man nun die linke Maustaste los, so werden alle Objekte, die sich vor dem Loslassen der linken Maustaste im Fangbereich befunden haben, markiert. Markierte Blöcke werden grau dargestellt, markierte Zwischenpunkte von Verbindungslinien rot. Verschiebt man einen markierten Block, so werden alle markierten Objekte mitverschoben. Klickt man mit der Maustaste in die leere Arbeitsfläche oder verschiebt man einen Zwischenpunkt, so werden alle Objekte entmakiert. Blöcke können auch markiert werden, indem man die linke Maustaste drückt während man die **STRG**-Taste gedrückt hält, während sich der Mauszeiger über dem Beschriftungsfeld eines Blockes befindet.

Modus	Cursor über	Cursorsymb.	Linke Maustaste	Rechte Maustaste
	Eingangskonnektor		Verbinden (bzw. Verbindungspkt. Mark.)	_
	Ausgangskonnektor		Verbinden (bzw. Verbindungspkt. Mark.)	Ausgang für Aufzeichnung markieren
	Beschriftungsfeld		*Drücken:* Block wird selektiert/deselektiert. *Drücken und Halten:* Block bewegt sich mit Cursor mit.	Lokales Blockmenü; handelt es sich um einen Beschriftungsblock, wird der Beschriftungsblockdialog angezeigt
	Verbindungslinie		Zwischenpunkt einfügen	_
	Zwischenpunkt		*Drücken und Halten:* Zwischenpunkt bewegt sich mit Cursor mit.	_
	Arbeitsfläche		*Drücken und Halten:* Fangbereich wird durch Ziehen des Mauszeigers erstellt. Nach Loslassen der linken Maustaste werden alle Objekte, die sich im Fangbereich befinden, markiert.	Lokales Arbeitsflächenmenü
	Arbeitsfläche, während Konstr. einer Verbindung		Zwischenpunkt setzen	_
	Untere rechte Ecke eines Blocks		*Drücken und Halten:* Blockgröße wird durch Ziehen des Mauszeigers verändert	_
	Ein- oder Ausgangskonnektor, Verbindungslinie, Zwischenpunkt		Das jeweilige Objekt wird gelöscht. (Bei Blöcken auch alle angeschlossenen Verbindungslinien!)	_
	Ein- oder Ausgangskonnektor		Es werden nacheinander zwei Konnektoren angeklickt. Die am ersten Konnektor angeschlossene Verbindungslinie wechselt zum zweiten Konnektor.	_

Abbildung 3.3: Betriebsarten, Cursorsymbole und Mausfunktionen

3.1.4 Dialoge

3.1.4.1 Blockauswahl Dialog

Mit diesem Dialog (Abbildung 2.3) werden Blöcke eingefügt und die Blockbibliothek gewartet. ANA bietet Blöcke aus verschiedenen Blockbibliotheken an. Jede Bibliothek ist in Gruppen unterteilt um gleichartige Blöcke zusammenzufassen. Durch Anklicken von Elementen der Listen „Bibliothek:“ und „Gruppe:“ kann man die Bibliotheken durchsehen. Durch Doppelklicken auf einen Eintrag in der Liste „Block:“ fügt man diesen Block in die Schaltung ein. Mit dem Button „**OK**“ wählt man ebenfalls den gerade markierten Block aus.Mit den Buttons „**Bearbeite BFL ...**“ oder „**Bearbeite Block ...**“ startet man einen eingebauten Editor um die Gruppendatei einer Bibliothek oder ein Blockskript zu lesen oder zu bearbeiten.

Bibliotheken:

Bibliotheken werden aufgrund eines Eintrags in der Datei `ANA.INI` bereitgestellt. Der folgende Ausschnitt aus einer `ANA.INI` Datei

```
...
BLOCKLIBS
   LIB ANA1.x_KompLib. AT C:\ANA2\ana1x
   LIB Misc_1 AT C:\ANA2\misc1
   LIB Local AT c:\ANA2\temp\local
ENDBLOCKLIBS
...
```

zeigt außer der speziellen Bibliothek „Local“ die Standardbibliotheken „ANA1.x_Komplib“ und „Misc1“. In "Local" dürfen keine selbsterstellten Blöcke abgelegt werden, da diese Wizardblöcke enthält und sie daher beim Laden und Speichern eines Beispiels automatisch gefüllt beziehungsweise geleert wird.

Eine Bibliothek in eine Ansammlung von ASCII Dateien in einem Verzeichnis. Ein eindeutiger Name für die Auswahlliste und die Angabe eines Pfades in `ANA.INI` genügen zur Installation einer Bibiliothek. Die Konfigurationsdatei `ANA.INI` wird nur beim Starten von ANA ausgewertet.

Gruppen:

Jede Bibliothek verfügt über ein Inhaltsverzeichnis, das in der Datei „`ABLOCKS.BFL`“ als ASCII Datei abgelegt ist:

```
...
$[Testsignale]
ASPRUNG,Sprungfunktion
ARAMPE,Rampenfunktion

$[Lineare Systeme]
AADD2,Addierer mit 2 Eingaengen
AADD3,Addierer mit 3 Eingaengen
...
```

Die Unterteilung in Abschnitte erfolgt mit „$[*Gruppenname*]“. Jeder Verweis auf einen Block beginnt mit dessen Dateinamen gefolgt von dessen Erläuterungstext. Dieser Text wird in der Liste „Block“ angezeigt.

Mit dem Button „**Bearbeite BFL ...**“ wird ein eingebauter Editor zum Lesen oder Verändern einer BFL-Datei gestartet. Nach dem Verlassen dieses Editors wird die Datei neu eingelesen und Änderungen somit sofort (ohne Neustart von ANA) wirksam.

Blöcke:

Jedem Block ist ein Blockskript in der Sprache ANAmdl (ANA Model Description Language) zugeordnet. Dieses Blockskript besteht aus Teilen, die für die Anschlüsse und Parameter des Blocks zuständig sind und aus weiteren Teilen welche die Modelleigenschaften definieren (siehe Abschnitt 4.1).

Ändert man nur die Modelleigenschaften eines Blocks, so werden diese Änderungen sofort bei der nächsten Simulation eines Beispiels wirksam.

Achtung! Alle Blöcke der Bibliotheken (außer die in „Local“) werden von den Beispielen nur referenziert, d.h. wird ein solcher Block geändert, wirkt sich dies auf alle Beispiele aus, die diesen Block verwenden.

Ändert man Teile, die Anschlüsse oder Parameter des Blocks beschreiben, so muß der betroffene Block oder die Schaltung neu geladen werden, damit die Änderungen vom Schaltungseditor erkannt werden.

Mit dem Button „**Bearbeite Block ...**“ wird ein eingebauter Editor zum Lesen oder Verändern des Blockskripts gestartet.

Einfügen selbstgeschriebener Blöcke:

Es wir dringend empfohlen, eine eigene, persönliche Bibliothek anzulegen. Dazu erweitert man den Abschnitt „BLOCKLIBS“ in ANA.INI beispielsweise um

```
...
   LIB Misc_1 AT C:\ANA2\misc1
   LIB Privat AT C:\ANA2\privat
   LIB Local AT c:\ANA2\temp\local
...
```

Die Bibliothek „Privat“ wird durch das Anlegen des Verzeichnis „C:\ANA2\privat“ geschaffen. Dies hat mit Hilfe externer Programme zu erfolgen. Ein Neustart von ANA ist durchzuführen.

Um einen Block in eine bestehende Bibliothek einzufügen wird „ABLOCKS.BFL“ mit Hilfe des Buttons „**Bearbeite BFL ...**“ im darauffolgenden Editor bearbeitet.

Der folgende Eintrag definiert eine Gruppe „Dampfkessel“ und einen Block mit dem Dateinamen „TYP221“ samt der Beschriftung „225 Liter Kessel Bauart 221“.

```
 ...
$[Dampfkessel]
TYP221,225 Liter Kessel Bauart 221
...
```

Mit dem Button „**Bearbeite Block ...**“ kann im darauffolgenden Editor die Datei „TYP221“ bearbeitet werden. Beispielstexte können vorteilhaft mit Hilfe der

Zwischenablage transferiert werden.

3.1.4.2 Parameter Dialog

Diesen Dialog (Abbildung 2.4) erhält man über das lokale Blockmenü. Der Dialog enthält Einträge für:

- Eingänge
- Ausgänge
- Parameter

Die Werte in den Parameterfeldern können verändert werden. Dabei ist man nicht nur auf Zahlen beschränkt, sondern kann auch Ausdrücke (z.B. `T1 + 5*T3`) verwenden. Die Gültigkeit dieser Ausdrücke wird erst beim Übersetzen des Modells berücksichtigt, so daß etwaige Fehleingaben erst dann zu einer entsprechenden Meldung führen.

OK: Der Dialog wird verlassen und die Einstellungen werden übernommen.

Abbrechen: Der Dialog wird verlassen. Es werden keine Änderungen vorgenommen.

Hilfe: Es wird Information über den Block angezeigt.

3.1.4.3 Markierungs Dialog

Dieser Dialog (Abbildung 2.7) dient zum Markieren bzw. Entmarkieren von Signalen eines Blocks. Alle markierten Signale werden während eines Simulationslaufes gespeichert und im ANAoszi angezeigt.

Mit der Auswahlliste in der linken Dialoghälfte wählt man das betroffene Signal. Mit dem Button „**Ändern**“ kann der jeweils gegenteilige Markierungszustand des Signals hergestellt werden.

Markierte Signale werden durch einen Stern (*) in der ersten Spalte der Signalliste angezeigt.

Sind außer den Blockausgangssignalen zusätzliche Signale (blockinterne Signale) markiert, so wird dies nach dem Verlassen des Dialogs durch einen kleinen roten Punkt in der rechten oberen Ecke des Blocks signalisiert.

Der Typ eines Signals wird in Klammern angeführt:

`signal`	Blockausgangssignal, wie an den Ausgangskonnektoren beschriftet
`state`	Zustandsvariable
`der`	Ableitung einer Zustandsvariablen
`param`	Parameter
`var`	Variable

Mit Hilfe der Buttons „**Alles markieren**“ bzw. „**Alles entmarkieren**“ können alle Signale eines Blocks gleichzeitig behandelt werden.

Markierungen für Blockausgangssignale können wahlweise durch Benutzung dieses Dialogs oder direkt durch Anklicken des Ausgangskonnektors des Blocks mit der rechten Maustaste verändert werden.

3.1.4.4 Gestaltungs Dialog

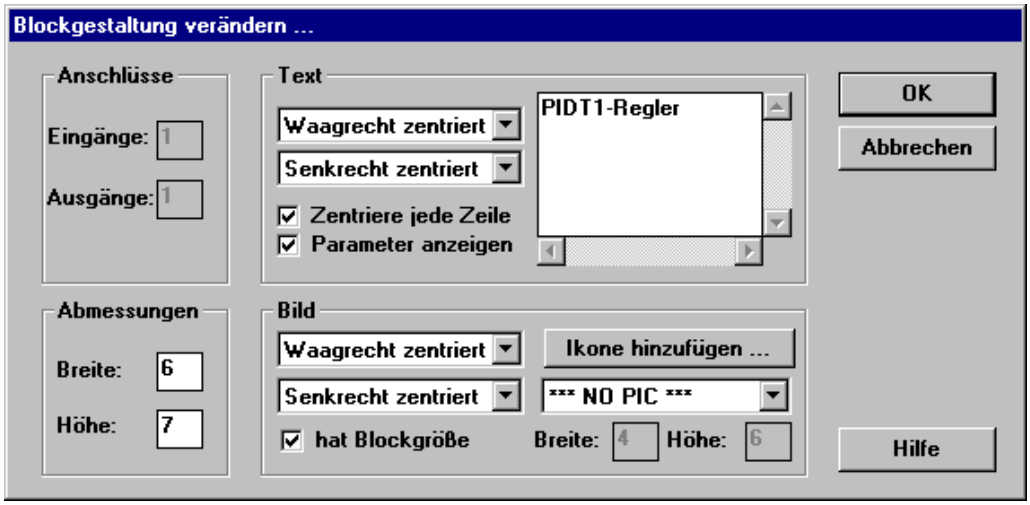


Abbildung 3.4: Dialog zur Blockgestaltung

Diesen Dialog (Abbildung 3.4) erhält man über das lokale Block-Menü (3.1.1.8). Er organisiert die optische Gestalt eines Blocks. Dabei können Abmessungen, Textinformation und Bildinformation vielfältig vorgegeben werden.

Abmessungen:

Die Abmessungen eines Blocks können entweder direkt mit der Maus, durch Fassen und Verschieben der *rechten unteren Ecke* eines Blocks oder durch Ändern der Einstellungen im Bereich „Abmessungen“ verändert werden. Die Maßeinheit für Breite und Höhe eines Blocks ist das Rastermaß des Schaltungseditors. In allen Fällen modifiziert ANA einen eingegebenen Vorschlag für die Blockabmessungen so, daß die Eingänge und Ausgänge des Blocks erkennbar bleiben. Daher werden im Anzeigebereich „Anschlüsse“ die Anzahl der Eingänge und Ausgänge des Blocks zur Information angezeigt.

Textinformation:

Text wird mit Hilfe eines Editierfeldes im Bereich „Text“ eingegeben. Neben dem Zentrieren jeder einzelnen Textzeile kann der Text auch noch durch Ankreuzen von „**Parameter anzeigen**“ um die Parameterliste des Blocks erweitert werden. Die Ausrichtung des Text relativ zum Blockrahmen kann eingestellt werden. Die Schriftart und Schriftgröße ist fix voreingestellt und kann nicht verändert werden. Ist die Textfläche größer als der Block, so wird die Textfläche an den Blockkanten durch Clipping beschnitten. Überlappen sich Textinformation und Bildinformation eines Blocks so wird der Text immer über das Bild geschrieben.

Bildinformation:

Blöcke können mit Bildern gestaltet werden. Dazu können entweder Dateien im ANAicl Format (Abschnitt 3.4) oder im Microsoft Bitmap Format (Dateierweiterung BMP) verwendet werden. Um eine Bilddatei verwenden zu können, muß sie im Verzeichnis das durch die `ANA.INI` Variable `ICONDIR` benannt wird, abgelegt sein. Unter dem Button „**Ikone hinzufügen**“ befindet sich eine Auswahlliste. Ist die gewünschte Grafik noch nicht in diesem Auswahlelement, so kann sie durch Betätigung des Buttons „Ikone hinzufügen“ in diese Liste übernommen werden. Die Abmessungen einer Grafik können entweder in der Einheit des Rastermaßes vorgeben werden oder es kann dies durch Ankreuzen von „**hat Blockgröße**“ automatisch mit der aktuellen Größe eines Blocks mitverändert werden. Dies kann auch dazu führen,

daß das Seitenverhältnis einer Grafik verloren geht und damit Verzerrungen entstehen. Durch einstellbare Ausrichtung der Grafik kann sich diese die Blocksymbolfläche mit einer Textinformation teilen (z.B. Textinformation „Oben bündig“, Bildinformation bei fixierten Abmessungen „Unten bündig“).

3.1.4.5 Signalnamen Dialog

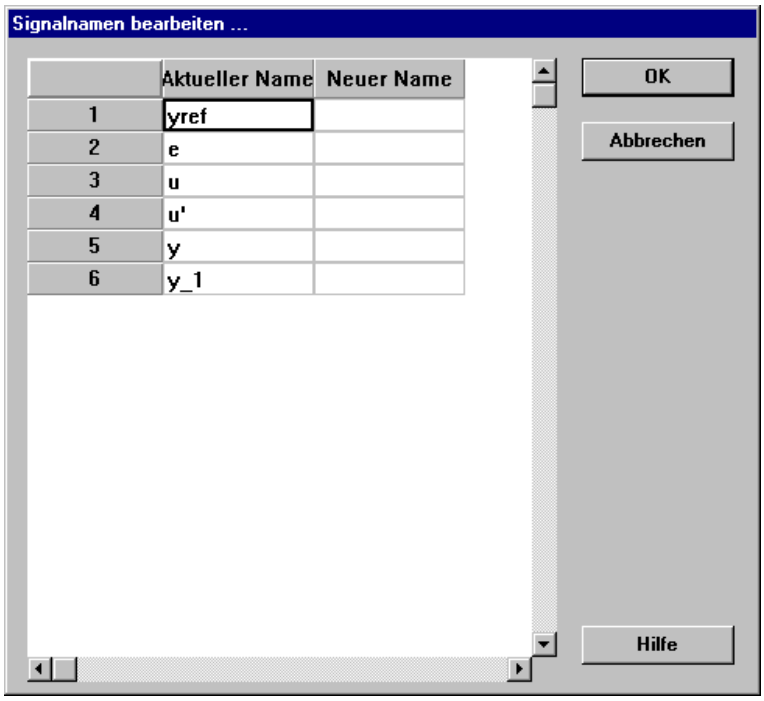


Abbildung 3.5: Dialog zum Editieren der Signalnamen

Dieser Dialog (Abbildung 3.5) wird über „**Bearbeiten → Signalnamen bearbeiten ...**“ aufgerufen. Um die Signalnamen zu ändern werden in der Spalte „**Neuer Name**“ die gewünschten neuen Namen eingetragen. Bleiben in dieser Spalte Einträge leer, so behält das betroffene Signal seinen (alten) Namen. Mit dem Button „**Abbrechen**“ kann ein Änderungsvorgang jederzeit verworfen werden.

*Bevor der Button „**OK**“ gedrückt wird, muß der Eingabemodus des zuletzt bearbeiteten Tabellenfeldes durch Drücken der Return-Taste abgeschlossen werden. Ansonsten wird die zuletzt getätigte Eingabe nicht mitverarbeitet. Eine abgeschlossene Eingabe erkennt man dadurch, daß die Eingabe in fetter Schrift dargestellt wird.*

Benennt man mehrere Signale mit dem gleichen neuen Namen, so wird man beim Verlassen des Dialoges gewarnt. In diesem Fall erhält nur das erste Signal den gewünschten Namen. Alle weiteren Signale werden mit einem veränderten Namen nach folgendem Schema benannt:

`Masse`	Gewünschter Name
`Masse_conflict`	1. Duplikat
`Masse_conflict'`	2. Duplikat
`Masse_conflict''`	3. Duplikat

3.1.4.6 Simulations Dialog

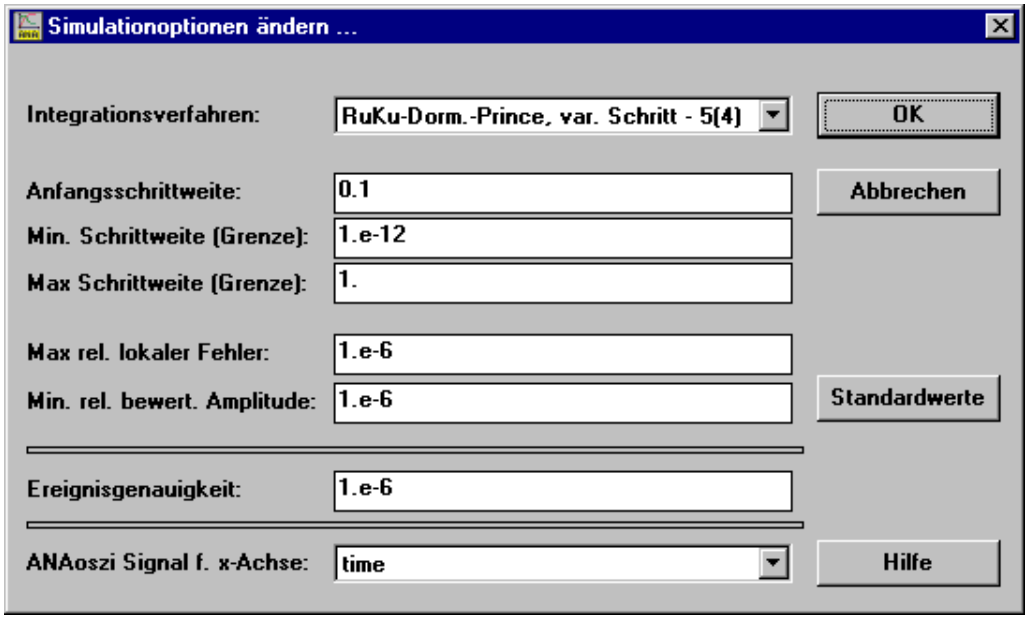


Abbildung 3.6: Dialog zum Einstellen der Simulationsparameter

Dieser Dialog (Abbildung 3.6) ermöglicht die Einstellung diverser Simulationsparameter und die Wahl des Signals für die ANAOszi x-Achse. Er wird über „**Simulation → Start ...**“ aufgerufen.

Mit diesem Dialog kontrolliert man:

- ANAOszi Plot x-Achsen-Steuerung
- Genauigkeitssteuerung
- Wahl eines Integrationsverfahrens

Nach erfolgter Übersetzung können hier die Parameter für den folgenden Simulationslauf eingestellt werden. Mit dem Button „**OK**“ startet man die Simulation. Alle Einstellungen in diesem Dialog werden beim Speichern einer ANA Schaltung mit abgelegt und stehen somit beim erneuten Laden einer Schaltung in der letzten Konfiguration zur Verfügung.

Der Button „**Standardwerte**“ bringt Grundeinstellungen für alle Parameter mit denen sich der Großteil aller Beispiele ohne weiteres Beachten der Bedeutung dieser Parameter simulieren läßt.

Der Dialog gliedert sich in zwei Aufgabenbereiche

- Wahl und Kontrolle des Integrationsverfahrens (oben und Mitte)
- Auswahl des Signals für die x-Achse des ANAOszi (unten)

ANAOszi Plots mit der Zeit als x-Achse:

Dazu wählt man den Eintrag „time“ in der Listbox. Alle für den Simulationslauf markierten Signale werden in Abhängigkeit der Zeit in einem gemeinsamen Koordinatensystem dargestellt. Während der Simulation kann der Maßstab des Graphen durch Bedienelemente des ANAOszi verändert werden.

ANAOszi Plots in Abhängigkeit eines Signales als x-Achse (Phasenebene):

Man wählt das gewünschte x-Achsen-Signal mit der Listbox. Damit läßt sich z.B. eine

Phasenebenendarstellung (2-dimensionaler Phasenraum, d.h. Darstellung der Ableitung eines Signale über dem Signal selbst) leicht erzielen, wenn sowohl das Signal als auch seine Ableitung in der Schaltung zur Aufzeichnung markiert wurden und als x-Achse das Signal gewählt wird.

Hinweis: Die Blockelemente der „ANA1.x_Komplib." stellen bei linearen Übetragungselementen in der Regel ein internes Signal yp zur Verfügung, welches die zeitliche Ableitung des Ausgangssignals y ist. Auch der ANA LaplaceWizard erzeugt nach Möglichkeit ein solches Signal yp passend zu y.

Genauigkeitssteuerung:

- Ereignisgenaugigkeit:
  Alle ANA Lösungsverfahren sind in der Lage „Time-Events" und „State-Events" mit einstellbarer Genauigkeit zu verarbeiten. Dazu wird die Lösungskurve des Integrationsverfahren gestückelt. Event-Zeitpunkte werden mit der Zeitunsicherheit ermittelt, die in dem Feld „**Ereignisgenaugigkeit**" eingestellt werden kann. Die Einheit der „Ereignisgenaugigkeit" sind Sekunden. Die minimale zulässige Schrittweite (Feld „**Min. Schrittweite (Grenze)**") muß kleiner als die geforderte Ereignisgenauigkeit sein, da der Simulator sonst beim Ermitteln eines Event-Zeitpunkts von der Schrittweitenüberwachung unterbrochen wird.

- Maximaler relativer lokaler Fehler und minimale relativ bewertete Amplitude:
  Bei allen Integrationsverfahren mit Schrittweitensteuerung (auch DASSL) kann die gewünschte Genauigkeit des Simulationsergebnisses direkt vorgegeben werden. Die aktuelle Schrittweite kann während der Simulation in der Statuszeile des ANAOszi überwacht werden. Sie wird dort mit dem Symbol h bezeichnet. Mit dem Feld „**Max. rel. lokaler Fehler**" kann der maximale relative Fehler eingestellt werden, den eine Lösungskurve bei der Berechung ihres nächstfolgenden Gitterpunktes erleidet. Relativ bedeutet, daß der absolute Amplitudenfehler durch die Augenblicksamplitude dividiert wird. Damit bei sehr kleinen Signalamplituden die Bewertung mit dem obigen relativen Fehlermaß zu keiner Übertreibung bei der Schrittweitenwahl führt, wird, wenn die Signalamplitude den im Feld „**Min. rel. bewert. Amplitude**" einstellten Amplitudenwert unterschreitet, der relative lokale Fehler durch Bezug auf diesen Wert ermittelt. Obwohl die einzelnen gleichzeitig auftretenden Signalamplituden sehr unterschiedlich sein können, kann in dieser Version nur ein Wert für alle Zustandsgrößen vorgegeben werden.
  Da oben beschriebene Bewertungsvorgang pro Integrationsvariable (Zustandsgröße) durchgeführt wird ist der eingestellte maximale relative lokale Fehler der Maximalwerte der einzelnen Fehler. D.h. es wird die Maximumnorm des Fehlers bewertet.

- Anfangsschrittweite:
  Bei Verfahren mit automatischer Schrittweitensteuerung kann in diesem Feld die Anfangsschrittweite eingestellt werden. Beim DASSL-Algorithmus wird diese ebenfalls automatisch ermittelt und dieses Feld ist ohne Funktion. Bei Verfahren mit fester Schrittweite entspricht der hier als „Anfangsschrittweite" eingestellte Wert gleichzeitig der festen Schrittweite mit der die gesamte Simulation (mit Ausnahme der Event-Behandlung) durchgeführt wird.

- Minimale Schrittweite (Grenze):
  In diesem Feld wird die kleinste zulässige Schrittweite angegeben. Bei dem Unterschreiten dieses Wertes bricht der Simulator ab. Wird dieser unterschritten, so ist entweder

  - das Lösungsverfahren zu schlecht – oder

- das Differentialgleichungssystem zu steif – oder
- es wurde eine Unsteigkeit nicht als Event implementiert – oder
- die minimale Schrittweite größer als die Ereignisgenaugkeit – oder
- das Problem benötigt zur Lösung eben kleinere Schrittweiten.

- Maximale Schrittweite (Grenze):
  Vor allem hochwertige Integrationsverfahren vergrößern die Schrittweite zwischen den Gitterpunkten der Lösung sehr schnell. Mit Hilfe des Feldes „Max. Schrittweite (Grenze)“ wird die Schrittweite auf diesen eingestellten Wert begrenzt. Dies geschieht ohne Rückmeldung.

Wahl eines Integrationsverfahrens:

ANA stellt folgende Lösungsverfahren zur Verfügung:

- Einschrittverfahren der Runge-Kutta Klasse:
  Sie können für die Lösung nicht-steifer (keine stark unterschiedliche Eigenwerte) Differentialgleichungsysteme eingesetzt werden. Steife Differentialgleichungssysteme (mit stark unterschiedlichen Eigenwerten) können mit diesen Verfahren nur unter hohem Zeitaufwand gelöst werden. In einem solchen Fall sollte der DASSL Algorithmus versucht werden.

  - Runge-Kutta Dormand-Prince 5(4) mit variabler Schrittweite:
    Dies ist das Standardverfahren. Die Lösung wird durch Anstückelung eines Polynoms fünfter Ordnung gebildet. Zur Schrittweitenkontrolle wird mit der Lösung durch ein Polynom vierter Ordnung verglichen.

  - Runge-Kutta Dormand-Prince 8(6) mit variabler Schrittweite:
    Dieses Verfahren bietet bei nichtsteifen (siehe oben) Diffentialgleichungssystemen eine hohe Genauigkeit bei geringer Schrittweite. Da aber der Rechenaufwand pro Schritt steigt ist eine Zeitersparnis nur dann bemerkbar, wenn die Zeitdistanz der Visualisierung im ANAOszi (Kommunikationsintervall) nicht zu gering gewählt wurde und die Schaltung nicht durch ständige Ereignisbehandlung dominiert wird.

  - Runge-Kutta Fehlberg 2(3) mit variabler Schrittweite:
    Runge-Kutta Fehlberg 4(5) mit variabler Schrittweite:
    Runge-Kutta England 4(5) mit variabler Schrittweite:
    Runge-Kutta 4(5) mit variabler Schrittweite:
    Sie sind für Vergleichszwecke vorhanden.

- Euler Integration mit fixer Schrittweite (1. Ordnung):
  Runge-Kutta 4 mit fixer Schrittweite:
  In diesem Fall ist der Wert im Feld „Anfangsschrittweite“ der unveränderbare Wert der Simulationsschrittweite. Eine Ausnahme bilden zeitgesteuerte (Time-Events) und zustandsgesteuerte Ereignisse (State-Events). Solche werden immer durch systematische Kontrolle der Schrittweite mit der eingestellten „Ereignisgenauigkeit“ nach dem Verfahren der Bisektion ermittelt. Die Genauigkeit der Lösungskurve kann nur indirekt durch Variation der fixen Schrittweite beeinflußt werden.

- DASSL-Algorithmus:
  Das behandelte Differentialgleichungssystem wird durch den Einsatz dieses impliziten Lösungsverfahrens auch dann zeitlich attraktiv gelöst, wenn die Schaltung stark unterschiedliche Eigenwerte aufweist (d.h. das Differentialgleichungssystem steif ist). Der DASSL-Algorithmus wurde für die korrekte Behandlung von Time-Events und State-Events adaptiert. Die

Verarbeitung von impliziten Differentialgleichungen wird von ANAmdl in dieser Version nicht unterstützt. Bei nichtsteifen System sind die meisten Einschrittverfahren der Runge-Kutta Klasse bei gleicher Genauigkeit schneller.

3.2 ANAOszi

Das ANAOszi dient zur Aufzeichnung der markierten Signale. Diese werden bereits während der Simulation aufgezeichnet, wodurch der Simulationsfortschritt mitverfolgt werden kann. Die Farben der Signale, der Maßstab für die *x*- und *y*-Achse können eingestellt, das Ergebnis kann gedruckt, als MATLAB (`*.m`-File) oder ASCII-File gespeichert beziehungsweise in das Clipboard kopiert werden.

3.2.1 Menüs

Den Zusammenhang zwischen den einzelnen Menüpunkten und den Buttons in der Werkzeugleiste zeigt Abbildung 3.7.

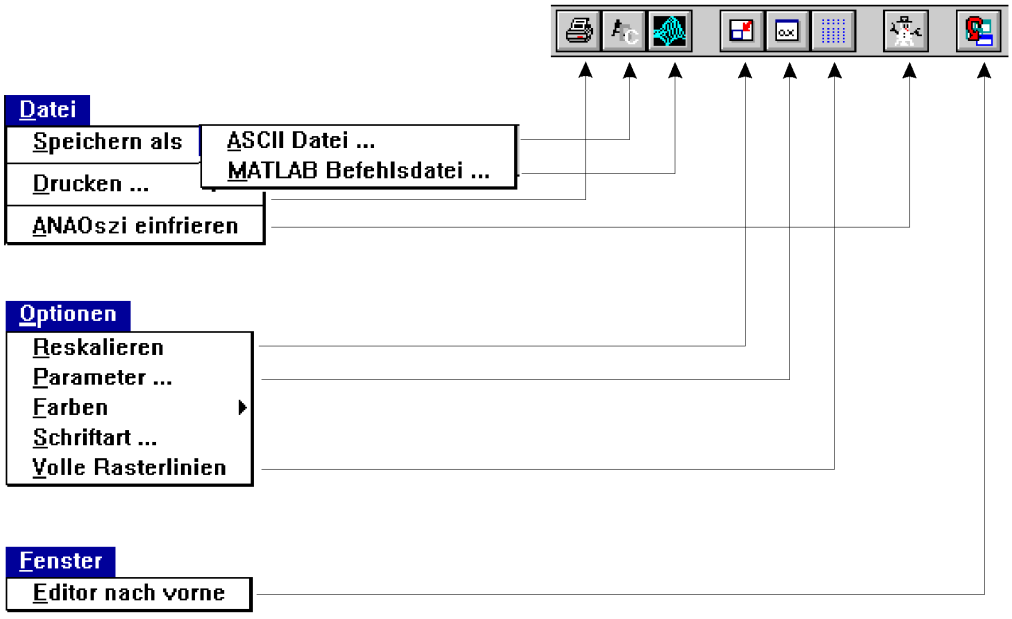


Abbildung 3.7: Menüs und dazugehörige Werkzeugleiste des ANAOszi

3.2.1.1 Datei Menü

Speichern als → ASCII-Datei: Die Daten werden als ASCII-File abgespeichert, dessen Name über einen Datei Dialog festgelegt wird.

Speichern als → MATLAB Befehlsdatei: Die Daten werden als `*.m` Datei abgespeichert. Über einen Datei Dialog wird der Name der `*.m` Datei festgelegt. In MATLAB stehen die Daten als Vektoren zur Verfügung. Außerdem wird durch Aufrufen der `*.m` Datei das Oszi-Bild in MATLAB gezeichnet.

Drucken : Der aktuelle Bildausschnitt wird ausgedruckt. Die Größe des Gitters ist unabhängig von der Auflösung des Druckers 1cm x 1cm bzw. 1,5cm x 1,5cm (siehe Abschnitt 3.2.2.1).

ANAOszi einfrieren: Wird dieser Menüpunkt ausgewählt, so wird das ANAOszi-Fenster eingefroren, das heißt, daß beim erneuten Starten einer Simulation ein neues Fenster geöffnet wird und das alte Fenster unverändert bleibt. Es können bis zu neun ANAOszi-Fenster eingefroren werden.

3.2.1.2 Bearbeiten Menü

In die Zwischenablage kopieren: Der aktuelle Bildausschnitt wird in das Clipboard kopiert.

3.2.1.3 Optionen Menü

Reskalieren: Der Maßstab wird automatisch so ermittelt, daß das ganze Bild sichtbar und die Beschriftung „günstig“ ist.

Parameter : Es wird der Parameter-Dialog (Abschnitt 3.2.2.1) aufgerufen.

Farben: Über die Untermenüpunkte „**Hintergrund**“, „**Raster**“ und „**Schrift der Skala**“ können die entsprechenden Farben eingestellt werden.

Schriftart: Es wird der Schriftarten-Dialog (Abschnitt 3.2.2.2) aufgerufen.

Volle Rasterlinien: Durch Auswahl dieses Menüpunktes wird das Koordinatengitter durchgezogen gezeichnet, ansonsten gepunktet.

3.2.1.4 Fenster Menü

Editor nach vorne: Das Editor-Fenster wird in den Vordergrund geholt.

3.2.2 Dialoge

3.2.2.1 Parameter Dialog

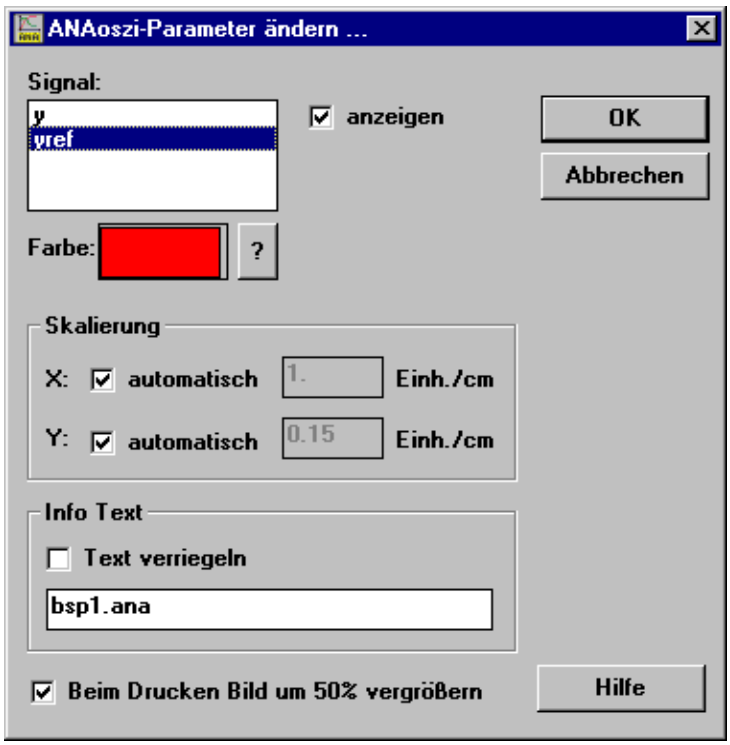


Abbildung 3.8: Parameter Dialog des ANAOszi

Mit diesem Dialog (Abbildung 3.8) kontrolliert man wichtige Eigenschaften des ANAOszi Bildes:

- Auswahl von Signalen für die Anzeige
- Anzeigefarbe von Signalen
- Skalierung der Achsen
- Beschriftung des Diagramms
- Größe des Diagramms beim Ausdruck (nur hier änderbar!)

Auswahl und Farbe der Signale:

Die aufgezeichneten Signale sind in der Liste „**Signal**“ wählbar. Für den markierten Listeneintrag zeigt die Checkbox „**anzeigen**“ ob dieses Signal in Diagramm erscheint. Standardmäßig erscheinen alle aufgezeichneten Signale.

Im Feld „**Farbe**“ erscheint die zugeordnete Signalfarbe. Drückt man den Button „**?**“ erscheint ein Farbauswahldialog. Man kann dem Signal eine andere Farbe zuordnen. Diese Einstellungen werden beim Speichern eines Beispiels nicht gesichert.

Skalierung von Achsen:

Das ANAOszi teilt x- und y-Achse immer in 10 Skalenteile. Ein Skalenteil hat beim Standardausdruck entweder 1 cm oder bei 50% Vergrößerung 1,5 cm Länge. Mit den beiden Checkboxen wird, getrennt für x- und y-Achse, zwischen automatischer Skalierung und manueller Skalierung umgeschaltet.

Bei automatischer Skalierung werden alle aufgezeichneten Signale zur Berechnung des Maßstabs miteinbezogen. Die Checkbox „**anzeigen**“ hat hier keine Auswirkung. Das Eingabefeld für einen Skalenmaßstab wird erst bei deaktivierter automatischer Skalierung aktiv. Eine Verkleinerung der Skalierungszahl (gemessen in Einheit/Skalenteil, d.h. Einheit/cm bei Standarddruck) bewirkt eine Vergrößerung des Bildes.

Ist ein Bild größer als die durch die Maßstäbe eingestellte Anzeigefläche, so erscheinen Rollbalken mit deren Hilfe ein Bildausschnitt gewählt werden kann.

Die Letzteinstellungen der Skalierung werden beim Sichern eines Beispiels aufgezeichnet.

Tip: Vergrößern und Wählen eines Ausschnitts kann rasch und direkt mit der Maus durchgeführt werden (Abschnitt 3.2.3).

Beschriftung eines Diagramms:

Ein Diagramm besitzt einen Informationstext in der oberen Statuszeile des ANAOszi. Der Text ist standardmäßig der Dateiname der simulierten Schaltung und soll bei gleichzeitiger Anzeige mehrerer ANAOszi die Möglichkeit einer primitiven Dokumentation der Experimente ermöglichen. Deshalb kann dieser Text mit Hilfe dieses Dialogs verändert werden. Wählt man zusätzlich mit Hilfe der Checkbox die Option „**Text verriegeln**“ so bleibt dieser Text auch nach einer Neusimulation bestehen.

Ändern der Größe des Audrucks:

Der Skalenmaßstab für den Ausdruck des ANAOszi kann von 1 cm pro Skalenteil auf 1,5 cm pro Skalenteil vergrößert werden. Alle Ausdrucke des Diagramms werden mit Hilfe des Druckertreibers exakt auf diesen Maßstab berechnet. Die Checkbox „**Beim Drucken Bild um 50% vergrößern**“ ist standardmäßig angekreuzt.

3.2.2.2 Schriftarten Dialog

Abbildung 3.9: Schriftarten Dialog des ANAOszi

Über diesen Dialog (Abbildung 3.9) kann für die Beschriftung Schriftart („**Gestalt**“) und Schriftgröße („**Größe**“) ausgewählt werden. Diese Einstellungen werden für beide Skalen und für die Signallegende verwendet. Sie werden nicht gespeichert.

3.2.3 Bildausschnitt vergrößern

Die Skalierung kann nicht nur über den Parameterdialog geändert werden, sondern es kann direkt mit der Maus ein Bildausschnitt vergrößert werden. Dazu drückt man die linke Maustaste und hält sie gedrückt. Bewegt man den Mauszeiger, so wird ein strichliertes Rechteck angezeigt, das den *neuen Bildausschnitt* darstellt. Läßt man nun die linke Maustaste los, so wird der durch das Rechteck markierte Bildausschnitt auf

volle Fenstergröße vergrößert. Es gibt keine direkte Möglichkeit zum vorherigen Bild zurückzukehren.

3.3 Der LaplaceWizard

Zusätzlich zu den über Blockbibliotheken einfügbaren Blöcken können Blöcke nun auch über den LaplaceWizard eingefügt werden. Der LaplaceWizard wird über das lokale Arbeitsflächenmenü (Abschnitt 3.1.1.7) aufgerufen wodurch der in Abbildung 3.10 dargestellte Dialog geöffnet wird.

Mit Hilfe dieses Dialogs können Übertragungsfunktionen als Blöcke mit einem Eingang und einem Ausgang erstellt werden. Dazu wird die gewünschte Übertragungsfunktion als Ausdruck angegeben.

Wird innerhalb des Ausdrucks der Laplace Operator *s* verwendet, so erhält man ein kontinuierliches Übertragungssystem.

Wird stattdessen der Operator *z* ($z = e^{sT}$) verwendet, so definiert man ein zeitdiskretes Übertragungssystem und muß für die so definierte Differenzengleichung noch die Abtastzeit T als Parameter festlegen. Der Ausgang solcher Systeme zeigt zwischen den Abtastzeitpunkten ständig den letzen Ausgabewert. Dies entspricht dem Verhalten eines Abtast-Halteglieds nullter Ordnung (Gh0).

s-Operator und *z*-Operator dürfen in einem Ausdruck nicht gemischt vorkommen. Der Grad des Zählerpolynomns (höchste Potenz des Operators im Zähler) muß kleiner oder gleich dem Grad des Nennerpolynoms (höchste Potenz des Operators im Nenner = Ordnung des Systems) sein. Ansonsten ist das System entweder ideal differenzierend (bei Differentialgleichungen) oder nicht kausal (bei Differenzengleichungen).

Es stehen folgende Elemente zu Verfügung:

s	Laplace-Operator (Heavyside-Operator) → Differentialgleichung
z	Operator der Form $z = e^{sT}$ → Differenzengleichung
j	Imaginäre Einheit ($\sqrt{-1}$)
T	Abtastzeit (Dieser Parameter ist nur bei Definition von G(z) vorbelegt.)
( )	Klammern zur Steuerung der Auswertereihenfolge
+, -, *, /	Grundrechnungsarten; Das Minuszeichen dient auch als Vorzeichen
^	Potenzieren eines Ausdrucks mit einem ganzzahligen Exponenten

Alle anderen Namen, die mit einem Buchstaben beginnen werden als Parameter interpretiert und können später über den Parameterdialog verändert werden.

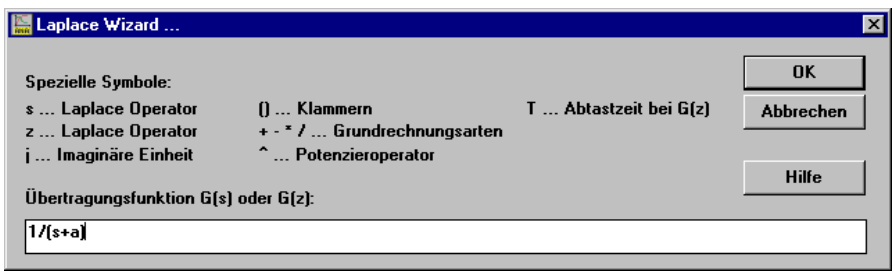


Abbildung 3.10: Der Laplace-Wizard

Beispiele:

Gegeben: $G(s) = k\dfrac{1+sT_v}{1+sT_d}$ ... (PDT_1)

Eingabe: `k*(1+s*Tv)/(1+s*Td)`

Gegeben: $G(s) = k_r + \dfrac{k_n}{s} + k_d\dfrac{s}{1+sT_d}$ ... ($PIDT_1$)

Eingabe: `kr + kn/s + kd*s/(1+s*Td)`

Gegeben: $G(s) = \dfrac{3(s+2)^2(s+4)}{2s^3+\gamma s^2+1}$ ... ($PDD_2\ T_3$)

Eingabe: `3*(s+2)^2*(s+4)/(2*s^3+gamma*s^2+1)`

Gegeben: $G(z) = \dfrac{\dfrac{1}{z-1}+z^2+2}{z(2z^2-(z+2)^2)}$

Eingabe: `(1/(z-1) + z^2 + 2)/(z*(2*z^2 - (z+2)^2))`

Gegeben: $G(z) = \dfrac{(z-1)(z+0{,}5+j0{,}5)(z+0{,}5-j0{,}5)}{(z-1)^3}$

Eingabe: `(z-1)*(z+0.5+0.5*j)*(z+0.5-0.5*j)/(z-1)^3`

Gegeben: $G(z) = k$

Eingabe: `k*z/z`

Hinweis: Durch den Einsatz des *z*-Operators (der so wirkungslos ist) wird ein zeitdiskretes System erzeugt.

3.4 ANAicl

3.4.1 Koordinaten

Der Ursprung des Koordinatensystems befindet sich links oben. Alle Koordinatenwerte sind Pseudokoordianten, d.h. sie können beliebig gewählt werden und haben nichts mit der tatsächlichen Größe der Ikone zu tun. Syntaktisch gilt für Koordinatenwerte:

koordwert := digit { digit };
digit := 0 ... 9

Der Referenzpunkt für Ellipse und Rechteck befindet sich *in der Mitte* derselben.

3.4.2 Befehle

3.4.2.1 MoveTo: `MT x,y`

Der Cursor wird an die Stelle (x, y) bewegt und (x, y) ist die neue aktuelle Position.

3.4.2.2 LineTo: `LT x,y`

Es wird von der aktuellen Position nach (x, y) eine Linie in der aktuellen Linienfarbe und im aktuellen Linientyp gezeichnet. (x, y) ist die neue aktuelle Position.

3.4.2.3 Ellipse: `EL a,b`

Es wird an der aktuellen Position eine Ellipse der Breite a und der Höhe b in der aktuellen Linienfarbe, dem aktuellen Linientyp und dem aktuellen Fülltyp gezeichnet. Die aktuelle Position wird nicht verändert.

3.4.2.4 Rectangle: `RT a,b`

Es wird an der aktuellen Position ein Rechteck der Breite a und der Höhe b in der aktuellen Linienfarbe, dem aktuellen Linientyp und dem aktuellen Fülltyp gezeichnet. Die aktuelle Position wird nicht verändert.

3.4.2.5 LineStyle: `LS type,width`

Setzt Linienbreite und Linientyp. Für *type* sind folgende Werte möglich:

1	voll
2	strichliert
3	punktiert
4	Strich-Punkt
5	Strich-Punkt-Punkt
6	unsichtbar

3.4.2.6 LineColor: `LC col`

Setzt die Linienfarbe. Für *col* sind folgende Werte möglich:

1	rot
2	grün
3	blau
4	schwarz
5	weiß
6	hellgrau
7	grau
8	dunkelgrau

3.4.2.7 FillStyle: `FS hatch,col`

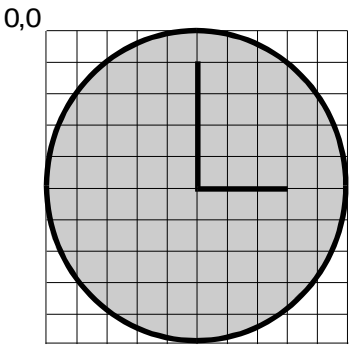
Setzt Füllmuster und Füllfarbe für folgende Objekte:

- Ellipse
- Rechteck

Für *col* sind die unter `LC` angegebenen Werte möglich. Zusätzlich existiert noch die Farbe `0`, d.h. das Innere des Objekts wird nicht gefüllt. Für *hatch* sind folgende Werte gültig:

1	4	7
2	5	
3	6	

Beispiel: Abbildung 3.11 zeigt jenes Icon, das beim ANA Steuerblock Verwendung findet.




```
LS 1,1;
FS 7,6;
LC 3;
MT 5,5;
EL 10,10;
LS 1,2;
LC 4;
MT 5,1;
LT 5,5;
LT 8,5;
```

Abbildung 3.11: Ein ANAIcon

4 Die Blockbeschreibungssprache ANAmdl

Die ANAModelDescriptionLanguage (ANAmdl) dient zwei Zwecken:

- Zum Einen ist sie eine Blockbeschreibungssprache, die zum Erstellen von benutzerspezifischen Teilsystemen („Blöcke“) dient. Blockskripts werden als ASCII-Datei erstellt und sind dann über die graphische Oberfläche von ANA ansprechbar. Das Erscheinungsbild solcher Blöcke ist durch das Hinzufügen von Grafikbefehlen in den Blockskripts beeinflußbar.
- Andererseits enthält die ANAmdl Anweisungen zum Verbinden von Teilsystemen zu einem Gesamtmodell. Der Benutzer der graphischen Oberfläche von ANA wird mit dieser Aufgabe nur indirekt konfrontiert, da er sein Modell „mit der Maus“ erstellt.

4.1 ANAmdl – Beschreibung von Blöcken

Es folgt hier keine formale Definition der ANAmdl sondern eine Einführung in Form einfacher Beispiele. Der syntaktische Aufbau der Sprache ist den Beispielen zu entnehmen.

4.1.1 Beispiel I: Addierer – 2 Eingänge

Mathematische Formulierung des Problems:

$$y(t) = k_1 u_1(t) + k_2 u_2(t)$$

Das dazugehörige Blockskript:

```
1     BLOCK AAdd2;
2     $ BLOCKDIM -1 -1 6 3
3     $ TEXT "Addierer"
4     $ TEXTALIG 1 1 1 1
5     $ ICONNAME "*** NO PIC ***"
6     $ ICONDIM 4 2
7     $ ICONALIG 1 1 1
8     INPUT
9          u1 "[1] Addierereingang";
10         u2 "[1] Addierereingang";
11    OUTPUT
12         y "[1] Addiererausgang";
13    PARAMETER
14         k1 = 1 "[1] Verstaerkung";
15         k2 = 1 "[1] Verstaerkung";
16    SIM
17         y = k1*u1 + k2*u2;
18    ENDSIM
19    ENDBLOCK AAdd2;
```

BLOCK

Eine Blockbeschreibung wird eingeleitet durch das Schlüsselwort `BLOCK` gefolgt von einem frei wählbaren Blocknamen. Beendet wird die Blockbeschreibung durch das Schlüsselwort `ENDBLOCK` gefolgt von dem bei `BLOCK` vergebenen Blocknamen. Alle Ausdrücke werden mit einem Strichpunkt abgeschlossen. Die Verwendung von Umlauten ist *nicht* gestattet.

INPUT,
OUTPUT,
PARAMETER

Die Zeilen 2 bis 7 beinhalten Informationen, die für die Darstellung des Blockes auf der Arbeitsfläche benötigt werden. Diese Grafikbefehle werden stets durch „$“ eingeleitet. Bei allen folgenden Beispielen werden sie nicht mehr angeführt.

Die Zeilen 8 bis 15 enthalten den *Interfaceteil* einer Blockbeschreibung. Er besteht aus maximal drei Abschnitten – den Deklarationen der Eingänge, Ausgänge und Parameter. Diese Abschnitte sind durch die Schlüsselwörter `INPUT`, `OUTPUT` und `PARAMETER` gekennzeichnet. In Zeile 9 und 10 werden zwei Eingänge, `u1` und `u2` deklariert. Die Zeichenkette `"[1] Addierereingang"` dient als Kommentar, der auf der graphischen Oberfläche eingeblendet wird. Er ist optional, sollte aber immer angegeben werden. Die Parameterdeklaration unterscheidet sich von der Deklaration der Ein- und Ausgänge dadurch, daß den Parametern Default-Werte zugewiesen werden können (Zeilen 14 und 15).

SIM

Die Zeilen 16 bis 18 enthalten den *Simulationsteil* , eingeleitet durch `SIM` und abgeschlossen durch `ENDSIM`. Zeile 17 stellt die eigentliche Modellbeschreibung dar – wie man sieht kann die mathematische Darstellung praktisch unverändert übernommen werden. Die angegebene Gleichung ist jedoch *keine* Zuweisung (wie bei sequentiellen Programmiersprachen üblich), sondern eine echte mathematische *Äquivalenzbeziehung*. Stehen zwischen `SIM` und `ENDSIM` mehrere Gleichungen, so werden diese automatisch sortiert um den Gesetzen der Kausalität zu genügen. Die Gleichungen sind in expliziter Form anzugeben, d.h. auf der linken Seite darf nur eine Variable stehen.

4.1.2 Beispiel II: PT1

Mathematische Formulierung des Problems:

Das Übertragungsverhalten eines Verzögerungsgliedes erster Ordnung wird durch die Übertragungsfunktion

$$F(s) = \frac{Y(s)}{U(s)} = \frac{k}{1 + sT_1} = \frac{k}{T_1} \cdot \frac{1}{1/T_1 + s}$$

beschrieben. Eine äquivalente Zustandsraumdarstellung in Regelungsnormalform lautet:

$$\begin{aligned} \dot{x} &= -\frac{1}{T_1} x + u \\ y &= \frac{k}{T_1} x \end{aligned}$$

Für die Anfangsbedingung soll gelten:

$$AB = y(t=0) = \frac{k}{T_1} x(t=0)$$

Das dazugehörige Blockskript:

```
1     BLOCK APT1;
2     INPUT
3          u "[1] PT1 Eingang";
4     OUTPUT
5          y "[1] PT1 Ausgang";
6     PARAMETER
7          k = 1.0 "[1] Verstaerkung";
8          T1 = 1.0 "[s] Zeitkonstante";
9          AB = 0.0 "[1] Anfangsbedingung";
10    STATE
11         x "[1] PT1";
12    SIM
13         x .= -1 /T1 *x +u;
14         y = k /T1 *x;
15    ENDSIM
16
17    INIT
18         x = AB *T1 /k;
19    ENDINIT
20    ENDBLOCK APT1;
```

STATE

Der Interface-Teil dieses Blockskripts ist analog zum ersten Beispiel aufgebaut. Neu ist jedoch das Schlüsselwort in Zeile 10. Nach STATE folgt die Deklaration der Zustandsvariablen. Die notwendige Anzahl von Zustandsvariablen richtet sich nach der Anzahl der Energiespeicher im System, oder ist – anders ausgedrückt – gleich der Ordnung des zugrundeliegenden Differentialgleichungssystems. Dieses Beispiel enthält eine Zustandsvariable, nämlich x.

.=

Der Simulationsteil unterscheidet sich nur wenig von der mathematischen Darstellung. Bloß der Ableitungspunkt über dem x ($\dot{x}$) wird zum Ableitungsoperator (.=). Da jedoch links von diesem Operator nur eine Variable stehen darf, liest sich Zeile 13 genauso wie ihre mathematische Entsprechung (x Punkt ist gleich ...).

INIT
ENDINIT

Neu ist auch der Initialisierungsteil, eingeschlossen von INIT und ENDINIT (Zeilen 17 bis 19). Alle Anweisungen in diesem Teil werden zum Startzeitpunkt der Simulation ($t = 0$) ausgeführt. In Zeile 18 wird die Anfangsbedingung gesetzt.

4.1.3 Beispiel III: Zweipunktregler

Formulierung des Problems:

Eine der Stärken der ANAmdl ist die Fähigkeit, unstetige Elemente exakt beschreiben zu können. Es wäre möglich, diese Unstetigkeiten nicht zu beachten –so wie in ANA1.x – und damit einen Fehler in Kauf zu nehmen, da der Integrationsalgorithmus somit nicht feststellen kann, wann der Wert einer Zustandsvariablen springt.

Welche Sprachkonstrukte notwendig sind, um unstetige Elemente zu beschreiben, soll anhand eines Zweipunktreglers mit Hysterese dargestellt werden. zeigt die Kennlinie dieses Reglers.

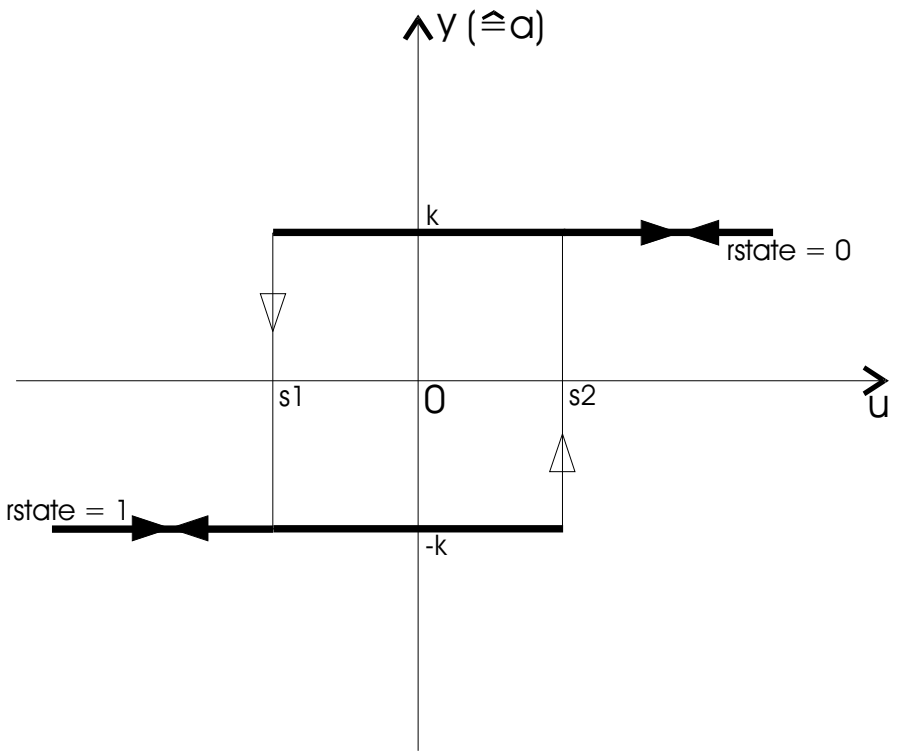


Abbildung 4.1:Kennlinie des Zweipunktreglers

Mit Hilfe der Zeichnung kann folgender Algorithmus formuliert werden:

---

wird (u > s2):
    setze y =k;
wird (u < s1):
    setze y = - k;
sonst:
    lasse y unverändert

Es gilt:

- Das Springen von *y* wird als Ereignis (event) bezeichnet. Das Eintreten dieses Ereignisses ist von *u* abhängig.
- Ein Ereignis tritt nur dann ein, wenn die jeweilige Bedingung (*u > s2*, *u < s1*) *zum ersten Mal* erfüllt ist nachdem sie *davor* nicht erfüllt war. Man muß also die *positive Flanke* der Bedingungen auswerten.

Das eben beschriebene Modell soll noch verfeinert werden. Um die maximale Schaltfrequenz des Reglers zu begrenzen, wird die Blockierzeit T_r eingeführt: Bevor der Regler seinen Schaltzustand wechseln kann, muß mindestens die Zeit T_r vergangen sein.

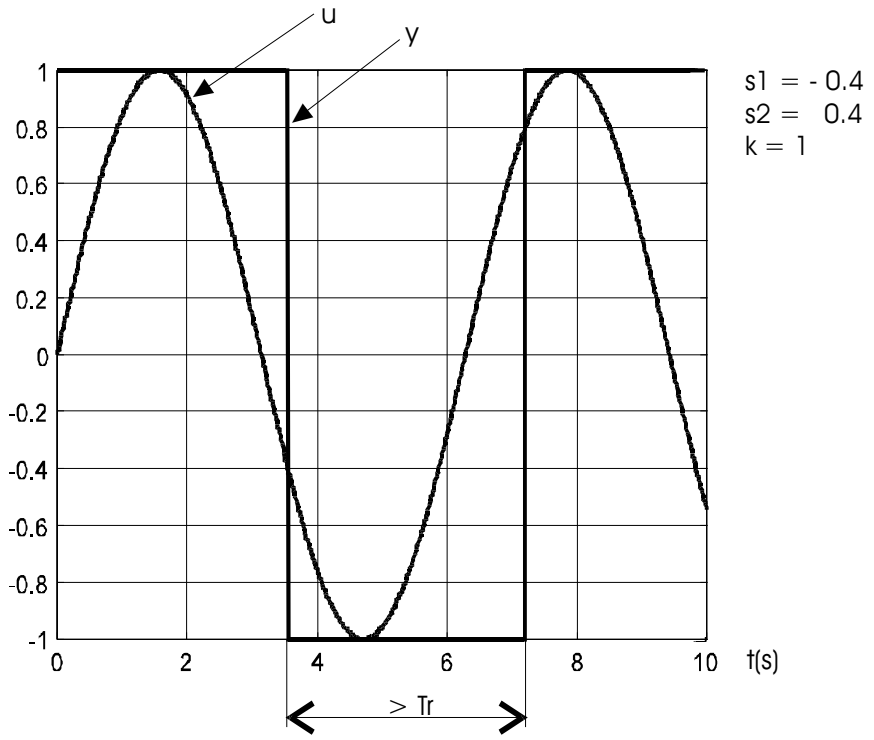


Abbildung 4.2: Blockierzeit T_r

In Abbildung 4.2 sieht man, daß der Regler aufgrund des Sinuseingangs bei der zweiten Vollwelle nicht aufgrund der Schaltschwelle, sondern erst nach abgelaufener Blockierzeit umschaltet.

Damit kann man den Algorithmus neu formulieren:

---

blockierzeit abgelaufen:
    wird (u > s2):
        setze y = k; starte blockierzeit;
    wird (u < s1):
        setze y =-k; starte blockierzeit;
    sonst:
        lasse y unverändert;
sonst:
    mache nichts;

Das Blockskript:

```
1     BLOCK AZPunkt;
2     INPUT
3          u "[1] Zweipunktregler Eingang";
4     OUTPUT
5          y "[1] Zweipunktregler Ausgang";
6     PARAMETER
7          k = 1.0 "[1] Stellgroesse";
8          s1 = -0.5 "[1] Ausschaltschwelle";
9          s2 = 0.5 "[1] Einschaltschwelle";
10         AB = 1 "[1] Anfangsbedingung";
11         Tr = 0.001 "[s] Umschaltzeit des Reglers";
12    VAR
13         a DISCRETE "Reglerausgang";
```

```
14          rstate DISCRETE "Zweiwertiges Stellrelais";
15          waitstate DISCRETE "Wartezustand";
16     SIM
17          SWITCH waitstate
18          CASE 0:
19               SWITCH rstate
20               CASE 0:
21                    ONRISE u <s1 DO set_to_neg;
22               CASE 1:
23                    ONRISE u >s2 DO set_to_pos;
24               DEFAULT:
25                    EXIT "Unbek. Zustand fuer rstate";
26               ENDSWITCH
27          CASE 1:
28               # Nothing
29          DEFAULT:
30               EXIT "Unbekannter Zustand fuer
waitstate";
31          ENDSWITCH
32          y = a <- set_to_pos(u) <- set_to_neg(u);
33     ENDSIM
34
35     PROCEDURE set_to_pos;
36          a = k;
37          rstate = 0;
38          waitstate = 1;
39          SCHEDULE wake2punkt AT NOW+Tr;
40          STORE ALL;
41     ENDPROCEDURE
42
43     PROCEDURE set_to_neg;
44          a = -k;
45          rstate = 1;
46          waitstate = 1;
47          SCHEDULE wake2punkt AT NOW+Tr;
48          STORE ALL;
49     ENDPROCEDURE
50
51     PROCEDURE wake2punkt;
52          waitstate = 0;
53          STORE;
54     ENDPROCEDURE
55
56     INIT
57          IF s2 < s1 THEN
58               EXIT "s2 darf nicht kleiner als s1
sein!";
59          ENDIF
60          waitstate = 0;
61          rstate = 0;
62          IF AB >= 0 THEN
63               a = k;
64               rstate = 0;
65          ENDIF
66          IF AB < 0 THEN
67               a = -k;
68               rstate = 1;
69          ENDIF
70     ENDINIT
71     ENDBLOCK AZPunkt;
```

VAR
DISCRETE

Der Interface-Teil dieses Blockskripts ist analog zu den vorangegangenen Beispielen aufgebaut. Neu ist unter anderem das Schlüsselwort VAR in Zeile 12, das zum Deklarieren von Variablen dient. Diese sind hier zusätzlich mit dem optionalen Bezeichner DISCRETE versehen. Für Discrete-Variablen gilt:

Im SIM - ENDSIM Teil dürfen sie *nur* auf der rechten Seite einer Gleichung stehen.

In Prozeduren dürfen sie auch auf der linken Seite einer Anweisung stehen.

Sie müssen im INIT - ENDINIT Abschnitt initialisiert werden.

Betrachtet man den Simulationsteil (Zeilen 16 bis 33) so findet man hier den vorhin beschriebenen Algorithmus verwirklicht. Die Variable *waitstate* dient zur Realisierung der Blockierzeit, *rstate* gibt den Schaltzustand des Reglers wieder:

waitstate	0	Regler kann schalten
	1	Regler blockiert
rstate	0	Regler steht auf $+k$ $(a = k)$
	1	Regler steht auf $-k$ $(a = -k)$

SWITCH
ONRISE

Die Variable *a* kennzeichnet den Zustand des Reglerausgangs. Die Zustände der Variablen *rstaite* und *waitstate* werden über einen SWITCH-Verteiler abgefragt. Die flankenbehaftete Abfrage boolscher Ausdrücke erfolgt durch das Schlüsselwort ONRISE. Zeile 21 wird gelesen als: „Wenn *u* zum ersten- mal kleiner wird als *s1* dann rufe die Prozedur set_to_neg auf“.

<-

In Zeile 32 wird *y* mit der Discrete-Variablen *a* gleichgesetzt. *a* ist jedoch abhängig vom Zustand von *u* und wird dementsprechend in den Prozeduren set_to_pos und set_to_neg gesetzt. Diese Abhängigkeit wird durch den Depend-Operator (<-) ausgedrückt.

PROCEDURE

Prozeduren werden durch das Schlüsselwort PROCEDURE gefolgt vom Prozedurnamen und einem Strichpunkt eingeleitet und mit ENDPROCEDURE abgeschlossen. Alle dazwischen liegenden Anweisungen werden *sequentiell* abgearbeitet. Hier dürfen Zuweisungen an Discrete-Variablen stattfinden.

SCHEDULE
NOW

In den Zeilen 39 und 47 trifft man auf die SCHEDULE-Anweisung. „SCHEDULE wake2punkt AT NOW+Tr“ heißt, daß die Prozedur wake2punkt zum Zeitpunkt „jetzt plus T_r“ aufgerufen wird. Damit ist die Blockierzeit realisiert: Sowohl in set_to_pos als auch in set_to_neg wird *waitstate* = 1 gesetzt und der Aufruf der Prozedur wake2punkt für T_r-Sekunden später eingeplant. Solange *waitstate* gleich 1 ist, verzweigt der SWITCH-Verteiler in die Zeile 27 wo „nichts“ passiert. (Das Zeichen # in Zeile 28 leitet einen Kommentar ein. Alle nachfolgenden Zeichen

in der selben Zeile werden ignoriert.) Wird nun `wake2punkt` nach T_r-Sekunden aktiv, so wird *waitstate* auf 0 gesetzt (Zeile 52) und das Relais ist wieder schaltbereit.

`STORE`
Durch die Anweisung `STORE` (Zeile 53) werden markierte Signale (Die Markierung erfolgt in der graphischen Oberfläche, siehe Abschnitt 2.2) zu diesem Zeitpunkt abgespeichert. Durch `STORE ALL` (Zeilen 40 und 48) werden sowohl rechts- als auch linksseitiger Grenzwert aufgezeichnet, d.h. die Amplituden vor und nach der Schalthandlung.

`IF`
In Prozeduren und dem Initialisierungsabschnitt steht das `IF - THEN - EN-DIF`-Konstrukt zur Realisierung von Verzweigungen zur Verfügung (siehe Zeile 57 - 59).

4.1.4 Beispiel IV: Abtast-Halteglied

Formulierung des Problems:

Ein Abtast-Halteglied ist ein diskretes System, bei dem zu jedem Abtastzeitpunkt (nT) der Wert des Eingangssignales auf den Ausgang übertragen wird. Zwischen den Abtastzeitpunkten ändert sich der Wert des Ausgangssignales nicht. Mit der Abtastzeit T erhält man:

$$\begin{aligned} y(nT) &= u(nT) \\ y((n-1)T < t < nT) &= y(n-1)T \end{aligned}$$

Das Blockskript:

```
1     BLOCK ASundH;
2     INPUT
3          u "[1] S&H Eingang";
4     OUTPUT
5          y "[1] S&H Ausgang";
6     PARAMETER
7          T = 1.0 "[s] Abtastzeit";
8     VAR
9          y0 DISCRETE;
10    SIM
11         y = y0 <- sample(u);
12    ENDSIM
13
14    PROCEDURE sample;
15         y0 = u;
16         SCHEDULE sample AT NOW+T;
17         STORE ALL;
18    ENDPROCEDURE
19
20    INIT
21         y0 = 0;
22         SCHEDULE sample AT NOW;
23    ENDINIT
24    ENDBLOCK ASundH;
```

Die Discrete-Variable `y0` speichert den Zustand des Ausganges. Im Simulationsteil wird der tatsächliche Ausgang `y` mit `y0` gleichgesetzt (Zeile 11). Natürlich muß man angeben, wo und wodurch `y0` verändert wird. Dies geschieht, wie in Beispiel III,

durch den <--Operator.

Die Prozedur `sample` plant sich durch die `SCHEDULE`-Anweisung in Zeile 16 alle *T*-Sekunden selbst ein und überträgt zu diesen Zeitpunkten den Eingang *u* auf `y0`. Damit wird das vorhin beschriebene Verhalten erreicht.

4.1.5 Grapfikbefehle

Die Grafikbefehle beinhalten Informationen, die zur Darstellung des Blocks auf der Arbeitsfläche benötigt werden. Diese Befehle stehen immer unmittelbar nach `BLOCK <blockname>;`. Jeder Befehl wird durch `$` eingeleitet. Es müssen immer alle folgenden Befehle in derselben Reihenfolge angegeben werden:

```
$ BLOCKDIM <left> <top> <width> <height>
$ TEXT "<text>"
$      "Text Zeile zwei"
$ TEXTALIG <halign> <valign> <linecenter> <parashow>
$ ICONNAME "<filename>"
$ ICONDIM <width> <height>
$ ICONALIG <halign> <valign> <autosize>
```

Die Grafikbefehle spiegeln imwesentlichen die Einstellungsmöglichkeiten des Gestaltungs-Dialogs (Abschnitt 3.1.4.4) wieder.

`BLOCKDIM`: Mit *left* und *top* wird die Position der linken oberen Ecke angegeben, mit *width* und *height* die Breite bzw. Höhe des Blocks. In der Regel sollte für *left* und *top* immer -1 angegeben werden – damit wird der Block an der Stelle des Mauszeigers plaziert.

`TEXT`: Mit *text* wird jener Text angegeben, der im Beschriftungsteil des Blockes erscheinen soll. Einen Zeilenumbruch erhält man wie oben durch „Text Zeile zwei“.

`TEXTALIG`: Mit *halign* wird die horizontale Ausrichtung des Textes angegeben (0 . . .linksbündig, 1 . . . zentriert, 2 . . . rechtsbündig), mit *valign* die vertikale Ausrichtung (0 . . . oben, 1 . . . zentriert, 2 . . . unten). Mit *linecenter* wird jede Zeile innerhalb des Textblockes zentriert, mit *parashow* werden im Textbereich des Blockes die Parameter angezeigt (bei einem vorhandenen Textblock werden sie an diesen angefügt).

`ICONNAME`: *filename* enthält den Namen des Files, in dem das Icon mittels der ANAIcl beziehungsweise als Bitmap-File definiert ist.

`ICDONDIM`: Mit *width* wird die Breite und mit *height* die Höhe des Icons angegeben. Diese Einstellungen haben nur dann Wirkung, wenn bei `ICONALIG` autosize = *0* angegeben wird.

`ICONALIG`: Mit *halign* wird die horizontale Ausrichtung des Icons angegeben (0 . . .linksbündig, 1 . . . zentriert, 2 . . . rechtsbündig), mit *valign* die vertikale Ausrichtung (0 . . . oben, 1 . . . zentriert, 2 . . . unten). Mit *autosize = 1* wird die Größe des Icons automatisch an die Blockgröße angepaßt.

4.2 ANAmdl – Erstellen von Modellen

ANA verwendet die ANAmdl zum Speichern der Modelle. Es sollen die hierfür notwendigen Anweisungen der ANAmdl anhand des Beispiels aus Abschnitt 2 (Abbildung 2.5) erläutert werden: Das Modell wurde unter dem Namen `bsp1.ana` abgespeichert.

Hier das Listing:

```
1     CIRCUIT GedGen; # Bsp1.ana
2     USE asprung AS B_1
3     $ BLOCKLIB ANA1.x_KompLib.
4         A = 1;
5         T = 0;
6         A0 = 0;
7     $ BLOCKDIM 3 7 6 5
8     $ TEXT "Sprung"
9     $ TEXTALIG 1 1 1 1
10    $ ICONNAME "*** NO PIC ***"
11    $ ICONDIM 4 4
12    $ ICONALIG 1 1 1
13    $ SIGNAMES
14    $    "yref"
15    $ ENDSIGNAMES
16    ENDUSE
17    USE aadd2 AS B_2
18    $ BLOCKLIB ANA1.x_KompLib.
19        k1 = 1;
20        k2 = -1;
21    $ BLOCKDIM 12 9 6 3
22    $ TEXT "Addierer"
23    $ TEXTALIG 1 1 1 1
24    $ ICONNAME "*** NO PIC ***"
25    $ ICONDIM 4 2
26    $ ICONALIG 1 1 1
27    $ SIGNAMES
28    $    "e"
29    $ ENDSIGNAMES
30    ENDUSE
31    USE apid AS B_3
32    $ BLOCKLIB ANA1.x_KompLib.
33        k = 10;
34        kn = 0.5;
35        Tv = 0.0;
36        N = 10.0;
37    $ BLOCKDIM 20 7 6 7
38    $ TEXT "PIDT1-Regler"
39    $ TEXTALIG 1 1 1 1
40    $ ICONNAME "*** NO PIC ***"
41    $ ICONDIM 4 6
42    $ ICONALIG 1 1 1
43    $ SIGNAMES
44    $    "u"
45    $ ENDSIGNAMES
46    ENDUSE
47    USE apt1 AS B_4
48    $ BLOCKLIB ANA1.x_KompLib.
49        k = 1.0;
50        T1 = 2;
51        AB = 0.0;
52    $ BLOCKDIM 28 8 6 5
53    $ TEXT "PT1"
54    $ TEXTALIG 1 1 1 1
55    $ ICONNAME "*** NO PIC ***"
56    $ ICONDIM 4 4
57    $ ICONALIG 1 1 1
58    $ SIGNAMES
```

```
$    "u'"
$ ENDSIGNAMES
ENDUSE
USE apt1 AS B_5
$ BLOCKLIB ANA1.x_KompLib.
    k = 1.0;
    T1 = 0.1;
    AB = 0.0;
$ BLOCKDIM 36 8 6 5
$ TEXT "PT1"
$ TEXTALIG 1 1 1 1
$ ICONNAME "*** NO PIC ***"
$ ICONDIM 4 4
$ ICONALIG 1 1 1
$ SIGNAMES
$    "y"
$ ENDSIGNAMES
ENDUSE
USE btext15 AS B_6
$ BLOCKLIB Misc_1
$ BLOCKDIM 13 2 22 2
$ TEXT "Betragsoptimale Regelung"
$ TEXTALIG 1 1 1 1 2.00
$ ICONNAME "*** NO PIC ***"
$ ICONDIM 20 1
$ ICONALIG 1 1 1
$ SIGNAMES
$ ENDSIGNAMES
ENDUSE
USE acontrol AS B_7
$ BLOCKLIB ANA1.x_KompLib.
    te = 1.5;
    h = 0.01;
$ BLOCKDIM 3 16 6 5
$ TEXT ""
$ TEXTALIG 1 2 0 1
$ ICONNAME "control"
$ ICONDIM 2 2
$ ICONALIG 1 0 0
$ SIGNAMES
$    "-|"
$ ENDSIGNAMES
ENDUSE

CONNECT B_1.y TO B_2.u1;
$ INTERPOINTS
$ ENDINTERPOINTS
CONNECT B_2.y TO B_3.u;
$ INTERPOINTS
$ ENDINTERPOINTS
CONNECT B_3.y TO B_4.u;
$ INTERPOINTS
$ ENDINTERPOINTS
CONNECT B_4.y TO B_5.u;
$ INTERPOINTS
$ ENDINTERPOINTS
CONNECT B_5.y TO B_2.u2;
$ INTERPOINTS
$    10 11
$    10 15
```

```
119  $    44 15
120  $    44 10
121  $ ENDINTERPOINTS
122  SAVE  B_1.y B_5.y;
123
124  ENDCIRCUIT GedGen; # Bsp1.ana
125
126  /* BEGIN PARAMETER */
127  $ FNAME        nul
128  $ RMODE        0
129  $ HMODE        1
130  $ MODE         0
131  $ IALG         6
132  $ INITSTEP     0.1
133  $ MINSTEP      1.e-12
134  $ MAXSTEP      1.
135  $ ABSERR       1.e-6
136  $ RELERR       1.e-6
137  $ LIMIT        1.e-6
138  $ XVAR         0
139  /* END PARAMETER */
```

Durch `CIRCUIT` wird der Name des Modells festgelegt. Danach folgt eine Auflistung (Zeilen 2 bis 101) aller verwendeten Blöcke und die Festlegung, unter welchem Namen diese angesprochen werden sollen. Dazu dient die Anweisung „`USE <type> AS <blockname>`“. Danach wird mit „`$ BLOCKLIB <bibliotheksname>`“ die Blockbibliothek angegeben, aus der der Block entnommen wurde. Anschließend werden die Parameterwerte festgelegt, darauf folgen die Grafikbefehle (es werden die Standardvorgaben der Blockdefinition ersetzt – allerdings existieren noch zusätzlich Befehle zum Festlegen der Signalnamen, etwa Zeilen 13 bis 15). Die Blockinstanzierung wird durch `ENDUSE` abgeschlossen.

Danach werden die Verbindungen angegeben (Zeilen 103 bis 121). Diese Befehle haben die Form
„`CONNECT <blockname>.<ausgang> TO <blockname>.<eingang>`“.

Durch die Grafikbefehle „`INTERPOINTS - ENDINTERPOINTS`“ werden die Stützstellen der Verbindungslinien festgelegt. Mit `SAVE` (Zeile 122) werden die Signale, die aufgezeichnet werden sollen, angegeben (wieder in der Form `<label>.<sig>`). Ableitungen können durch Voranstellen von `d` angegeben werden.

Durch `ENDCIRCUIT` wird die Modellbeschreibung abgeschlossen.

Anschließend erfolgt eine Auflistung der über die Oberfläche getroffenen Einstellungen (Zeilen 126 bis 139). Diese Parameter sind intern.

4.3 Operatoren und Funktionen

4.3.1 Arithmetische Operatoren

Es stehen die binären Operatoren +, -, *, / sowie das unäre - zur Verfügung. Es gelten die in der Mathematik üblichen Vorangregeln. Durch Klammern kann die Auswertungsreihenfolge beeinflußt werden, z.B. `(4+3*5)*2 = 38`

4.3.2 Vergleichsoperatoren

==	Gleichheit
!= oder <>	Ungleichheit
<=	Kleiner oder gleich
>=	Größer oder gleich
<	Kleiner
>	Größer

4.3.3 Boolsche Operatoren

&& oder AND	Logisch Und
|| oder OR	Logisch Oder
! oder NOT	Logische Negation

Boolsche Ausdrücke werden nur soweit als notwendig ausgewertet.

4.3.4 Spezielle Operatoren

<-	Abhängigkeits (Depend) Operator
.=	Differential Zuweisungs-Operator
=	Äquivalenz Operator (in SIM – ENDSIM und INIT – ENDINIT)
:=	Zuweisungs Operator (in PROCEDURE – ENDPROCEDURE)

4.3.5 Funktionen und Prozeduren

4.3.5.1 Mathematische Funktionen

SQR(x)	x^2	COSH(X)	$\cosh x$
SQRT(x)	$\sqrt{x}$	TANH(X)	$\tanh x$
ABS(x)	$\|x\|$	ASINH	$\operatorname{asinh} x$
SIN(x)	$\sin x$	ACOSH	$\operatorname{acosh} x$
COS(x)	$\cos x$	ATANH	$\operatorname{atanh} x$
TAN(x)	$\tan x$	EXP(x)	e^x
ASIN(x)	$\arcsin x$	LN(x)	$\ln x$
ACOS(x)	$\arccos x$	LOG(x)	$\log x$
ATAN(x)	$\arctan x$	SIGN(x)	$\operatorname{sign} x$
POW(x,y)	x^y	ROUND(x)	□*x (aufrunden)*
SINH(X)	$\sinh x$	TRUNC(x)	□*x (abschneiden)*

4.3.5.2 Sonstige Funktionen und Prozeduren

4.3.5.2.1 RANDOMIZE()

Mit RANDOMIZE(AB) kann die Initialisierung des zentralen Zufallszahlengenerators eingestellt werden:

AB = 0 Initialisierung mit augenblicklicher Uhrzeit

AB > 0 Vorgabe des Initialisierungswertes als (long)AB d.h. als ganze positive Zahl zwischen 1 und 54773

4.3.5.2.2 RND()

Rauschen mittels gleichverteilter Zufallszahlen im Intervall [0 .. 1]. Siehe auch Block „Misc1\Rauschen\Gleichverteiltes Rauschen [0..1]“.

4.3.5.2.3 RNDEXP()

Rauschen mittels exponentiell verteilter Zufallszahlen im Intervall [0 ..∞]. Siehe auch Block „Misc1\Rauschen\Exponentiell verteiltes Rauschen [0..inf] p(x) = exp(-x) : x >= 0“.

4.3.5.2.4 RNDGAUSS()

Rauschen mittels gaussverteilter Zufallszahlen im Intervall [-∞ .. ∞]. Siehe auch Block „Misc1\Rauschen\ Gaussverteiltes Rauschen [-inf..inf] p(x) = 1/sqrt(2 Pi) exp(-x^2/2)“.

4.3.5.2.5 Filezugriff: FOPEN (), FREAD(), FGET(), FSTATUS(), FCLOSE()

Momentan können nur ASCII-Dateien *gelesen* werden. Die Befehle werden typisch in PROCEDURE-Abschnitten verwendet. Mit

```
h = FOPEN("c:\ana2\examples\test.dat");
```

wird der DISCRETE Variablen h ein Filehandle für die ASCII-Datei test.dat zugewiesen und bereits der erste Datensatz intern eingelesen. Jede Zeile der Datei bildet einen Datensatz, der mit FGET (siehe unten) gelesen werden kann. Innerhalb der Datei besteht ein Datensatz aus Fließkommazahlen, die durch Leerzeichen oder Tabulatoren zu trennen sind. Das Zeichen # leitet einen Kommentar bis Zeilenende ein. Auszug aus einer solchen Datei:

```
...
   0.22323   323.2323 23.223      # Das ist ein Kommentar
   12.32123  23.232 1.3
...
```

Mit z.B. IF FSTATUS(h) == 0 THEN ... kann geprüft werden ob ein Datensatz verfügbar ist. Mit x1 = FGET(h,1); wird die Zahl in der ersten Spalte, mit x2 = FGET(h,2); die Zahl in der zweiten Spalte ausgelesen. Der nächste Datensatz wird mit FREAD(h) angefordert. Einen Fall mittlerer Komplexität findet man im Beispiel FREAD.ANA.

4.3.6 Vordefinierte Konstanten

cPI oder PI	π	4.0*ATAN(1.0)
cE	e	EXP(1.0)

4.4 Reservierte Wörter

Folgende Wörter sind reserviert und dürfen nicht als Variablen-, Prozedur- oder Blocknamen verwendet werden:

ALL	DISCRETE	NOT	SOCKET
AND	DO	NOW	STATE
AS	ELSE	ONRISE	STORE
AT	ENDBLOCK	OR	STRING
BLOCK	ENDCIRCUIT	OUTPUT	SWITCH
BOOL	ENDIF	PARAMETER	THEN
BREAK	ENDINIT	Pi	THIS
CALL	ENDPROCEDURE	PI	TO
CASE	ENDSIM	PRIO	TYPENAME
cE	ENDSWITCH	PROCEDURE	USE
CIRCUIT	ENDUSE	QUIT	VAR
CONNECT	EXIT	SAVE	

cPi	FROM	SCHEDULE	
cPI	IF	SCHEDULE	
DEFAULT	INIT	SIM	
DELAY	INPUT	SLEEPUNTIL	

4.5 Kommentare

Alle Zeichen die nach # oder // in der selben Zeile folgen, ebenso alle Zeichen die zwischen /* und */ innerhalb einer Zeile eingeschlossen sind, werden ignoriert.

4.6 Verzeichnisstrukturen

Das Installationsprogram organsiert ANA in der in Abbildung 4.3 dargestellten Verzeichnisstruktur.

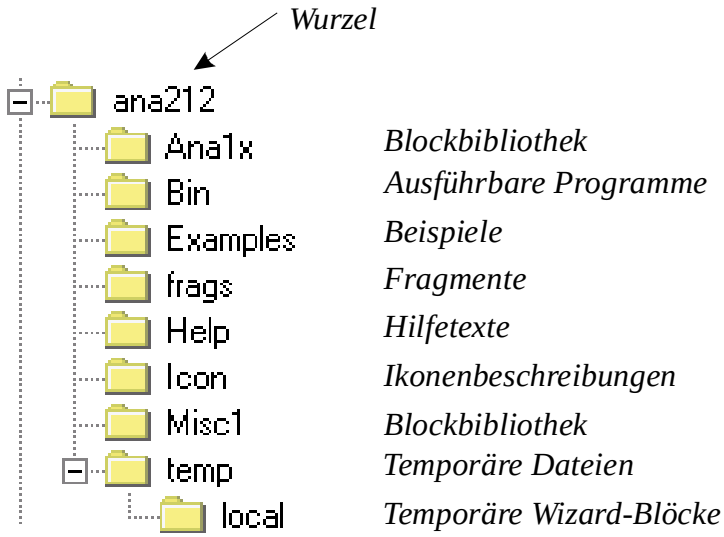


Abbildung 4.3: Verzeichnisstruktur

Diese Verzeichnisstruktur spiegelt sich im Initialisierungsfile ana.ini, das sich im Wurzelverzeichnis von ANA befindet, wieder. Dieses hat folgenden Aufbau:

anaini := *circuitdir fragdir tempdir icondir helpdir bindir blocklibs;*
circuitdir := ''CIRCUITDIR'' ''='' *verzeichnisname;*
fragdir := ''FRAGDIR'' ''='' *verzeichnisname;*
tempdir := ''TEMPDIR'' ''='' *verzeichnisname;*
icondir := ''ICONDIR'' ''='' *verzeichnisname;*
helpdir := ''HELPDIR'' ''='' *verzeichnisname;*
bindir := ''BINDIR'' ''='' *verzeichnisname;*
blocklibs := ''BLOCKLIBS'' *{ lib }*'' ENDBLOCKLIBS'' *;*
lib := ''LIB'' *libname* ''AT'' *verzeichnisname;*

Kommentare: Alle Zeichne, die nach # in der selben Zeile stehen, werden ignoriert.

Es gilt:

- Für CIRCUITDIR, FRAGDIR, ICONDIR, HELPDIR, BINDIR und TEMPDIR muß ein gültiger Verzeichnisname angegeben werden. CIRCUITDIR, FRAGDIR und TEMPDIR müssen schreibbar sein. Das Verzeichnis TEMPDIR wird von ANA intern verwendet. TEMPDIR wird beim Starten von ANA gelöscht.
- Es muß mindestens eine Blockbibliothek angegeben werden. Die Verzeichnisse, die mit LIB ... AT angegeben werden, müssen gültig sein und es muß sich ein Gruppenbeschreibungsfile mit demName ablocks.bfl in jedem dieser Verzeichnisse befinden.

Im Verzeichnis TEMPDIR muß sich ein schreibbares Unterverzeichnis Local befinden.

Beispiel für ana.ini:

```
#Standardverzeichnis für *.ana Dateien
CIRCUITDIR = C:\PROGRAMME\ana212\examples
#Verzeichnis für Fragmente
FRAGDIR = C:\PROGRA~1\ana212\frags
#Verzeichnis für temporäre Dateien
TEMPDIR = C:\PROGRA~1\ana212\temp
#Verzeichnis für Ikonenbeschreibungen
ICONDIR = C:\PROGRA~1\ana212\icon
#Verzeichnis für Hilfedateien
HELPDIR = C:\PROGRA~1\ana212\help
#Verzeichnis für ausführbare Programme
BINDIR = C:\PROGRA~1\ana212\bin

#Block Bibliotheken
BLOCKLIBS
   LIB ANA1.x_KompLib. AT C:\PROGRA~1\ana212\ana1x
   LIB Misc_1 AT C:\PROGRA~1\ana212\misc1
   LIB Local AT C:\PROGRA~1\ana212\temp\local
ENDBLOCKLIBS
```

Das Gruppenbeschreibungsfile `ablocks.bfl`

In jedem Verzeichnis, das einer Blockbibliothek zugeordnet ist, muß sich ein Gruppenbeschreibungsfile mit dem Namen `ablocks.bfl` befinden. Dieses hat folgenden Aufbau:

groupfile := { groupdescription };

groupdescription := ’’ `$` *’’ groupname { blockfilename ’’* `,` *’’ blocksymbname };*

Folgendes Beispiel zeigt einen Ausschnitt aus der ANA 1.x Kompatibilitätsbibliothek:

```
...
$[Testsignale]
ASPRUNG,Sprungfunktion
ARAMPE,Rampenfunktion
ASINUS,Sinusfunktion

$[Lineare Systeme]
AINTEG,Integrator
AADD2,Addierer mit 2 Eingaengen
AADD3,Addierer mit 3 Eingaengen
ATOTZEIT,Totzeit
APT1,PT1 - Verzoegerung 1. Ordnung
APT2,PT2 - Verzoegerung 2. Ordnung
ADIFF,Differenzierer

...
```

Den Zusammenhang zwischen `ana.ini` File, Gruppenbeschreibungsfile und Blockauswahldialog zeigt Abbildung 4.4. Siehe auch Abschnitt 3.1.4.1.

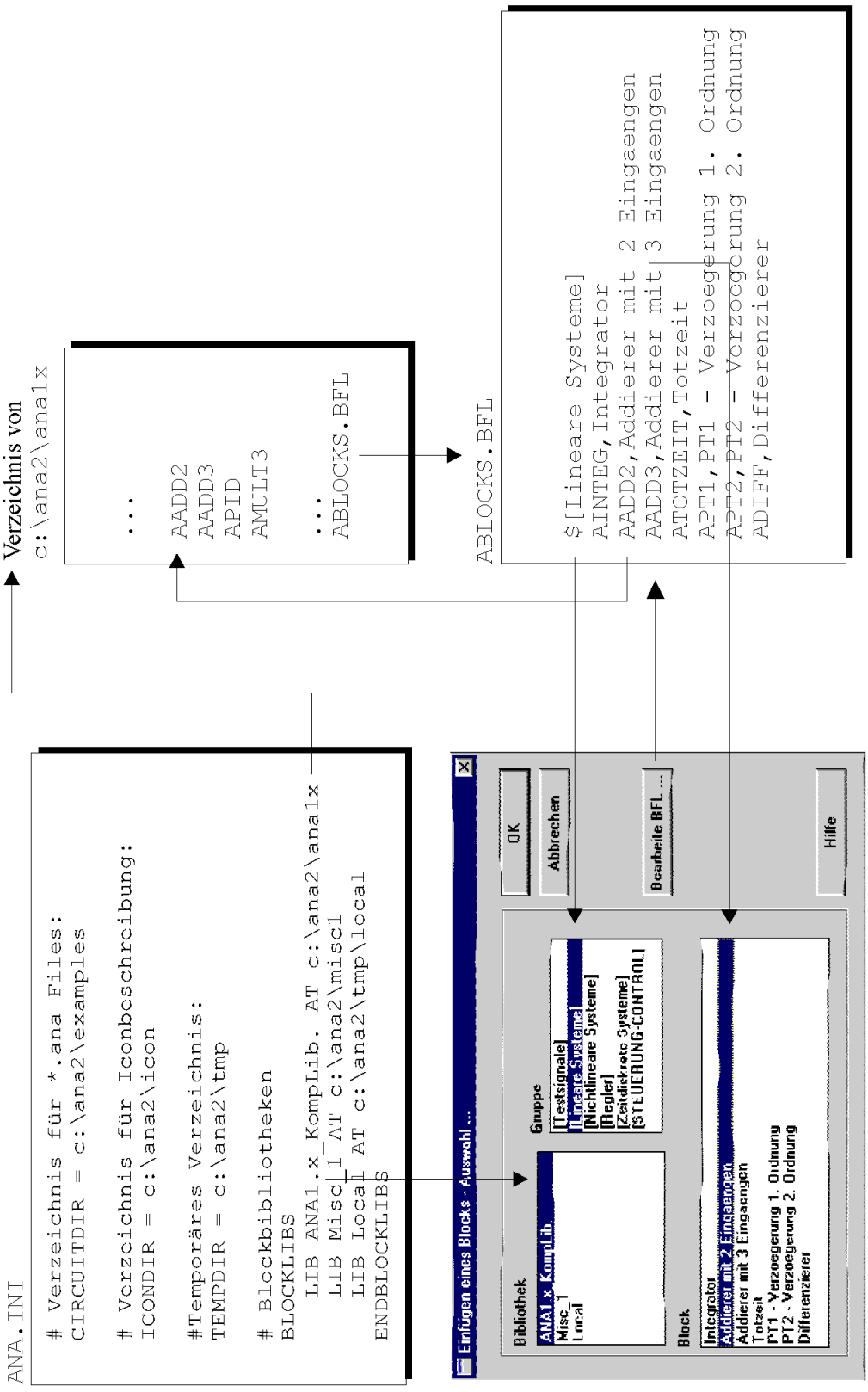


Abbildung 4.4: Zusammenhang zwischen ana.ini Datei, Gruppenbeschreibungsdatei und Blockauswahldialog