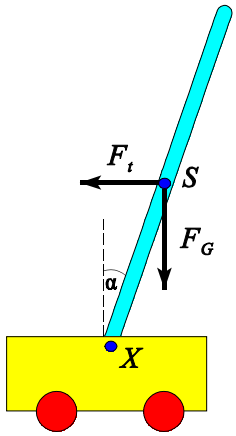
12 Anhang

12.1 Physik des invertierten Pendels

Laufkatze und invertiertes Pendel haben uns als simulierte Testobjekte für unsere Fuzzyprojekte gute Dienste geleistet. Für die Programmierung dieser Simulation, aber auch für ein tieferes Verständnis dieser Systeme muss man ihre Bewegungsgleichungen kennen. Vom physikalischen Standpunkt aus betrachtet sind die Laufkatze und das invertierte Pendel enge Verwandte. Ihre Bewegungsgleichungen unterscheiden sich lediglich in einem Vorzeichen. Aus diesem Grund soll hier auch nur eines von beiden, und zwar das invertierte Pendel, ausführlich behandelt werden.

Das System des invertierten Pendels besteht aus einem Stab, der mit einer Achse an einem Wagen befestigt ist. Auf den Wagen kann z.B. über einen Motor eine Kraft einwirken. Dadurch soll der Stab möglichst in seiner vertikalen Lage gehalten werden.



12.1 Kräfte am Pendel

Der Stab habe die Länge $2 \cdot l$; damit hat sein Schwerpunkt S den Abstand l zur Drehachse X. In diesem Schwerpunkt greift zunächst die Gewichtskraft F_G an. Sie führt zu einem Drehmoment

$$\begin{aligned} D_G &= F_G \cdot l \cdot \sin\alpha \\ &= m_S\, g\, l \sin\alpha \end{aligned} \tag{1}$$

Dabei bezeichnet m_S die Masse des Stabes. Wenn der Wagen nun in horizontaler Richtung beschleunigt wird, wirkt zusätzlich auf den Schwerpunkt S des Stabes eine Trägheitskraft $F_t = -m_S \ddot{x}$ entgegengesetzt zur Beschleunigungsrichtung; zu dem Drehmoment D_G kommt also noch ein weiteres Drehmoment hinzu:

$$\begin{aligned} D_t &= F_t \cdot l \cdot \cos\alpha \\ &= -m_S \ddot{x}\, l \cos\alpha \end{aligned} \tag{2}$$

Während man bei geradlinigen Bewegungen die Bewegungsgleichungen aus der Beziehung

$$\text{Masse} \cdot \text{Beschleunigung} = \text{Summe der angreifenden Kräfte}$$

$$m \cdot \ddot{s} = \sum_i F_i$$

erhält, gilt analog bei der Drehbewegung:

$$\text{Trägheitsmoment} \cdot \text{Winkelbeschleunigung} = \text{Summe der angreifenden Drehmomente}$$

$$\begin{aligned} \theta\, \ddot{\alpha} &= D_G + D_t \\ &= m_S\, l\, (g \sin\alpha - \ddot{x} \cos\alpha) \end{aligned} \tag{3}$$

Das Trägheitsmoment θ des Stabes errechnen wir an Hand der Abb. 12.2:

$$\begin{aligned}\theta &= \int r^2\, dm \\ &= \int_0^{2l} r^2\, \rho\, A\, dr \\ &= \left[\frac{r^3}{3}\rho A\right]_0^{2l} \\ &= \frac{(2l)^3}{3}\rho A \\ &= (2\cdot \rho\, l A)\,\frac{4}{3}l^2 \\ &= m_S\,\frac{4}{3}l^2\end{aligned} \tag{4}$$

Setzen wir dies in (3) ein, so ergibt sich:

$$m_S\,\frac{4}{3}\,l^2\,\ddot{\alpha} = m_S\,l\,(g\,\sin\alpha - \ddot{x}\,\cos\alpha)$$

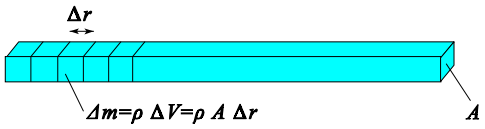
oder

$$\ddot{\alpha} = \frac{3}{4l}\,(g\,\sin\alpha - \ddot{x}\,\cos\alpha) \tag{5}$$

Auffällig ist, dass diese Gleichung nicht mehr die Masse m_S des Stabes enthält. Bedeutet dies, dass die Bewegung unseres Stabes unabhängig von dieser Masse ist? Zur Beantwortung dieser Frage muss geklärt werden, ob vielleicht $\ddot{x}$ von m_S abhängt. Leiten wir also auch für die Position x des Wagens eine Differenzialgleichung her: Von der Kraft $F(t)$, die zur Zeit t vom Motor auf den Wagen ausgeübt wird, führt ein Teil zur Beschleunigung des Wagens ($m_W\,\ddot{x}$), ein zweiter kompensiert die Reibungskraft ($c\,\dot{x}$), ein dritter verursacht eine Translationsbeschleunigung des Pendels. Der Rest führt zu der oben schon behandelten Drehbewegung des Pendels. Wir gehen jetzt davon aus, dass die Stabmasse m_S sehr viel kleiner ist als die Masse m_W des Wagens. In diesem Fall können die beiden letztgenannten Kraftwirkungen gegenüber den ersten beiden vernachlässigt werden:

$$F(t) = m_W\,\ddot{x} + c\,\dot{x} \tag{6}$$

Die Gleichungen (5) und (6) bilden ein Differenzialgeichungssystem für die Position x des Wagens und den Ablenkwinkel α des Stabes. Zusammen mit den Startwerten legt es die Bewegung unseres invertierten Pendels fest. Da die Stabmasse m_S weder in (5) noch in (6) auftaucht, ist sie für die Bewegung unseres Systems tatsächlich irrelevant, solange sie klein genug ist gegenüber der Wagenmasse m_W. Lediglich die Länge des Stabs beeinflusst seine Dynamik; die Winkelbeschleunigung ist um so größer, je kleiner die Stablänge ist. Dies erklärt auch, warum gerade bei kleinen Stablängen die Fuzzyregelung besonders sorgfältig konzipiert werden muss (vgl. Aufgabe 1 aus Kapitel 6).



12.2 Zur Berechnung des Trägheitsmoments

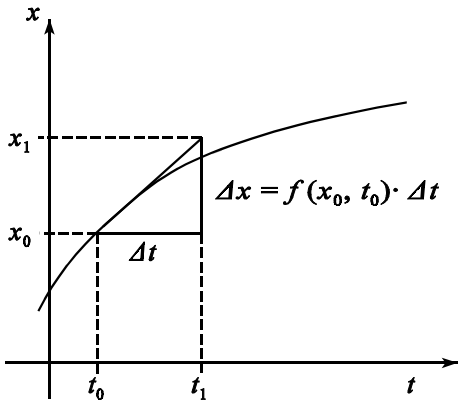
Für die Simulation unseres invertierten Pendels interessieren uns nur soche Bewegungen, bei denen der Stab mit geringen Amplituden um die labile Gleichgewichtslage $\alpha = 0$ schwingt. Wir können uns daher auf kleine Winkel α beschränken und die Gleichung (5) mit Hilfe der Beziehungen $\sin(\alpha) \doteq \alpha$ und $\cos(\alpha) \doteq 1$ nähern:[1]

$$\ddot{\alpha} = \frac{3}{4l}(g\alpha - \ddot{x}) \tag{7}$$

Die Gleichungen (6) und (7) bilden nun ein lineares Differenzialgleichungssystem zweiter Ordnung. Mit Einführung der Größen $v = \dot{x}$ und $\omega = \dot{\alpha}$ können wir es in ein Differenzialgleichungssystem 1. Ordnung für die vier Funktionen x, α, v und ω transformieren:

$$\begin{aligned} \dot{x} &= v \\ \dot{\alpha} &= \omega \\ \dot{v} &= \frac{F - cv}{m_w} \\ \dot{\omega} &= \frac{3}{4l}(g\alpha - \dot{v}) \end{aligned} \tag{8}$$

Für solche Differenzialgleichungssysteme gibt es zahlreiche Verfahren zur nummerischen Näherungslösung. Weil während der Simulation immer wieder regulierende Kräfte auf das System einwirken, brauchen wir in Hinblick auf langfristige Genauigkeit keine hohen Anforderungen an das Lösungsverfahren zu stellen; es genügt ein einfaches Verfahren, z.B. das Eulersche Polygonzugverfahren. Die nummerische Lösung einer Differenzialgleichung erster Ordnung $\dot{x} = f(x,t)$ mit dem Startwert $x(t_0) = x_0$ läuft nach diesem Verfahren folgendermaßen ab: Im x-t-Diagramm gibt $\dot{x}$ die Steigung des Graphens an; somit erhält man ausgehend vom Startwert x_0 als Näherungswert x_1 für einen um Δt späteren Zeitpunkt $t_1 = t_0 + \Delta t$:



12.3 Beim Eulerverfahren wird der Graph durch eine Tangente genähert.

$$\begin{aligned} x_1 &\doteq x_0 + \dot{x}(t_0)\cdot\Delta t \\ &= x_0 + f(t_0, x_0)\cdot\Delta t \end{aligned} \tag{9}$$

Um nun den x-Wert zum Zeitpunkt $t_2 = t_0 + 2\Delta t$ zu bestimmen, sehen wir x_1 als neuen Startwert an und erhalten analog:

$$x_2 = x_1 + f(t_1, x_1)\cdot\Delta t$$

[1] Das Programm *Fuzzy* benutzt ebenfalls diese Näherung. Für große Ablenkwinkel ist die Simulation deswegen ungenau. Wenn der Ablenkwinkel $\pi/2$ überschritten wird, ist ein Weiterführen der Simulation nicht mehr sinnvoll; das Programm bricht sie an dieser Stelle ab.

Allgemein ergibt sich der (i +1)-te Näherungswert aus dem i-ten Näherungswert mittels der Rekursionsformel

$$x_{i+1} = x_i + f(t_i, x_i)\cdot \Delta t\,, \tag{10}$$

wobei $t_i = t_0 + i\cdot \Delta t$ ist.

Im Prinzip ist dieses Verfahren um so genauer, je kleiner Δt gewählt wird. In der Praxis allerdings ist ein sehr kleines Δt nicht unbedingt günstig. In diesem Fall ist nämlich eine große Anzahl von Rechenschritten erforderlich; diese führt einerseits zu enormen Rechenzeiten, andererseits aber wegen der mit jedem Iterationsschritt verbundenen Rundungsfehlern auch zu erheblichen Ungenauigkeiten.

Unser invertiertes Pendel wird durch ein System von 4 gekoppelten Differenzialgleichungen erster Ordnung beschrieben. Ein solches System hat die allgemeine Form

$$\begin{aligned} \dot{x}_1 &= f_1(t, x_1, x_2, x_3, x_4) \\ \dot{x}_2 &= f_2(t, x_1, x_2, x_3, x_4) \\ \dot{x}_3 &= f_3(t, x_1, x_2, x_3, x_4) \\ \dot{x}_4 &= f_4(t, x_1, x_2, x_3, x_4) \end{aligned} \tag{11}$$

mit den Startwerten

$$\begin{aligned} x_1(t_0) &= x_{1,0} \\ x_2(t_0) &= x_{2,0} \\ x_3(t_0) &= x_{3,0} \\ x_4(t_0) &= x_{4,0} \end{aligned} \tag{12}$$

Wir können es wie im Fall einer einzigen Differentialgleichung lösen; dazu wenden wir bei jedem Iterationsschritt die Gleichung (9) auf alle 4 Variablen x_1, ..., x_4 an:

$$\begin{aligned} x_{1,i+1} &= x_{1,i} + f_1(t_i, x_{1,i}, x_{2,i}, x_{3,i}, x_{4,i})\cdot \Delta t \\ x_{2,i+1} &= x_{2,i} + f_2(t_i, x_{1,i}, x_{2,i}, x_{3,i}, x_{4,i})\cdot \Delta t \\ x_{3,i+1} &= x_{3,i} + f_3(t_i, x_{1,i}, x_{2,i}, x_{3,i}, x_{4,i})\cdot \Delta t \\ x_{4,i+1} &= x_{4,i} + f_4(t_i, x_{1,i}, x_{2,i}, x_{3,i}, x_{4,i})\cdot \Delta t \end{aligned} \tag{13}$$

Die x_r und die f_r (r = 1, ..., 4) kann man auch als Komponenten eines vierdimensionalen Vektors $\vec{x}$ bzw. $\vec{f}$ auffassen. Damit lassen sich die Gleichungen (11) bis (13) eleganter und übersichtlicher formulieren als

$$\begin{aligned} \dot{\vec{x}} &= \vec{f}(t, \vec{x}) \\ \vec{x}(t_0) &= \vec{x}_0 \\ \vec{x}_{i+1} &= \vec{x}_i + \vec{f}(t_i, \vec{x}_i)\cdot \Delta t \quad (t_i = t_0 + i\cdot \Delta t) \end{aligned} \tag{14}$$

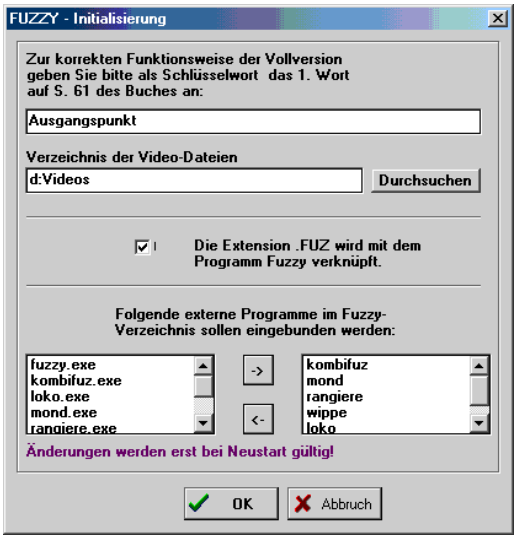
12.2 Installation und Grundsätzliches zur Bedienung von *Fuzzy*

Das Programm *Fuzzy* ist im Prinzip auf jedem IBM-kompatiblen Rechner mit WINDOWS 3.1 (oder höher) und 4MB RAM (oder mehr) lauffähig. Da die meisten Programmteile aber sehr viele Gleitkommaoperationen durchführen müssen, ist allerdings ein Coprozessor empfehlenswert. Insbesondere setzen die zeitkritischen Programmteile (Simulation) Taktfrequenzen von mindestens 25 MHz voraus.

Das Zusatzprogramm *RealExp*, welches zur Durchführung der Realexperimente erforderlich ist, benötigt einen IBM-kompatiblen Rechner mit WINDOWS 9x. Die Taktfrequenz sollte hier mindestens 233 MHz betragen.

Zur Installation legen Sie die CD in das CD-ROM-Laufwerk Ihres Computers und starten das Programm SETUP[1]. Alle benötigten Programme und Dateien werden nun in ein (Unter-)Verzeichnis der Festplatte entpackt. Standardmäßig heißt dieses Verzeichnis c:\Klett\Fuzzy; zu Beginn der Installation erhalten Sie jedoch die Möglichkeit ein anderes Verzeichnis zu wählen.

Wenn Sie jetzt das Programm *Fuzzy* zum ersten Mal starten, werden Sie aufgefordert, einige Angaben zur Initialisierung zu machen. Diese werden in der fuzzy.ini-Datei Ihres *Fuzzy*-Verzeichnisses gespeichert. Wenn Sie fuz-Dateien mit dem *Fuzzy*-Programm verknüpfen, kann *Fuzzy* auch über seine fuz-Dateien gestartet werden: Sobald Sie z.B. die Datei pendel.fuz im Dateimanager oder Explorer doppelt anklicken, wird das *Fuzzy*-Programm gestartet und diese Datei geladen. Das Herstellen dieser Verknüpfung ist der einzige Eingriff in Ihr WINDOWS-System, der bei der Installation bzw. Initialisierung vorgenommen wird.



12.4 Initialisierung des *Fuzzy*-Programms

Die auf der CD mitgelieferten Video-Dateien (über 100 MB) werden bei einer Standardinstallation nicht auf die Festplatte kopiert. Bei der Initialisierung können Sie aber das Verzeichnis, in dem diese Videos auf der CD liegen, angeben. Dazu klicken Sie auf die Taste “Durchsuchen” (s. Abb. 12.4), öffnen das Verzeichnis Video auf der CD und klicken eine der Videodateien an. Bestätigen Sie Ihre Eingabe jetzt mit OK. Von nun an kann das *Fuzzy*-Programm die Videos selbstständig finden und abspielen. Natürlich können Sie die Videos auch in ein Verzeichnis Ihrer Festplatte kopieren und dieses als Video-Verzeichnis festlegen.

Bei der Installation wurden auch einige weitere Simulationsprogramme in Ihr *Fuzzy*-Verzeichnis kopiert. Während der Initialisierung können Sie entscheiden, welche dieser Programme in das Simulation-Menü des *Fuzzy*-Programms eingebunden werden. Alle eingebundenen Programme können direkt vom *Fuzzy*-Programm gestartet werden. Bei einem solchen Aufruf übernehmen sie automatisch das aktuelle Projekt des *Fuzzy*-Programms, und zwar in der zuletzt abgespeicherten Version.

Die Funktion der einzelnen Komponenten des *Fuzzy*-Programms wird ausführlich in den einzelnen

[1] Wenn Sie das Betriebssystem Windows 3.X benutzen, öffnen Sie bitte das Verzeichnis Win31 auf der CD und starten Sie das Programm setup31.

Kapiteln beschrieben. Hier sollen nur einige grundlegende Aspekte angesprochen werden: Die einzelnen Programmfunktionen erreichen Sie in der üblichen Weise mit Hilfe der Menüzeile. Darüber hinaus steht Ihnen zur schnelleren Bedienung auch eine Knopfleiste zur Verfügung. Sobald sich der Mauszeiger über einem dieser Knöpfe befindet, erscheint in der Informationszeile eine kurze Erläuterung zur Bedeutung dieses Knopfes. Der Knopf wird betätigt, indem man ihn mit der linken Maustaste anklickt.

In der Regel erscheint dann ein zugehöriges Fenster, in welchem Sie die gewünschten Operationen durchführen können. Zum Verlassen der Fenster stehen der OK- und der Abbruch-Knopf zur Verfügung. Im Gegensatz zum OK-Knopf verfallen beim Abbruch-Knopf sämtliche Eingaben bzw. Änderungen, die in diesem Fenster vorgenommen worden sind.

Für die Eingabe von Fuzzyprojekten (Fuzzyvariablen und zugehörige Regeln) steht Ihnen ein einfacher Assistent zur Verfügung. Sie finden ihn im Hilfe-Menü.

Fuzzy besitzt eine umfangreiche Hilfedatei. Auf diese können Sie nicht nur über die Menüzeile, sondern auch von jedem Fenster aus zugreifen. Die Hilfestellung ist dabei kontextbezogen, d.h. Sie gelangen jeweils in denjenigen Abschnitt der Hilfedatei, der sich auf das aktuelle Fenster bezieht. Selbstverständlich können Sie jederzeit auch über ein Stichwortverzeichnis nach einem gewünschten Begriff suchen.

12.3 Programm-Interna von *Fuzzy*

Fuzzy wurde mit dem TURBO-PASCAL-Nachfolger DELPHI 1.0 programmiert. War unter TURBO-PASCAL die objektorientierte Programmierung nur ein von wenigen benutzter Zusatz, ist sie bei DELPHI unumgänglich. Tatsächlich handelt es sich bei allen Schaltern, Bildflächen und Textfenstern, kurz bei allen visuellen Komponenten, um Objekte. Selbst das Formular, auf dem diese plaziert sind, ist ein Objekt. Glücklicherweise ist der Umgang mit solchen Objekten wesentlich unkomplizierter geworden.

Objekte können Programme vereinfachen und übersichtlicher gestalten. Objekte zu definieren lohnt sich natürlich erst bei häufigem Einsatz. In unserem *Fuzzy*-Programm sind z.B. die Fuzzyvariablen als Objekte vom Typ `TFuzzy` realisiert. Dieser Objekttyp soll hier in einigen Zügen vorgestellt werden.

Zu einem Objekt gehören Eigenschaften und Methoden. Unser Objekt besitzt u.a. folgende Eigenschaften:

`a, b`:	linke und rechte Intervallgrenze der Grundmenge
`x`:	Wert der Grundmenge, für den die Zugehörigkeit bestimmt werden soll
`Anzahl`:	Anzahl der Terme
`Term[i,1], Term[i,2], ...`:	die charakteristischen Größen des i-ten Terms
`Zugehoerigkeit[i]`:	Zugehörigkeitsgrad von x zum i-ten Term
`Schwerpunktskoordinate`:	Schwerpunktskoordinate für das Gesamtdiagramm

Ist zum Beispiel `EingangA` ein Objekt vom Typ `TFuzzy`, so gibt `EingangA.Zugehoerigkeit[3]` den Zugehörigkeitsgrad von `EingangA.x` zum 3. Term an. Wie die Zugehörigkeiten berechnet werden, haben wir in Kapitel 3 kennengelernt. Hier sehen Sie nun den zugehörigen Programmausschnitt von `TFuzzy`. Lassen Sie sich dabei nicht verwirren: Allen Namen für die oben erwähnten Eigenschaften ist hier ein großes F vorangestellt.

```
function TFuzzy.GetZugehoerigkeit(i:integer):single;
begin
   if Not Fehler(i) then  {i-ter Term ohne Fehler?}
    if Fx<FTerm[i,1] then result:=0
    else if Fx<FTerm[i,2] then if FTerm[i,1]=FTerm[i,2]
      then result:=FTerm[i,5] else
         result:=FTerm[i,5]*(Fx-FTerm[i,1])/(FTerm[i,2]-FTerm[i,1])
    else if Fx<FTerm[i,3] then result:=FTerm[i,5]
    else if Fx<FTerm[i,4] then if FTerm[i,3]=FTerm[i,4]
      then result:=FTerm[i,5] else
        result:=FTerm[i,5]-FTerm[i,5]*
                                (Fx-FTerm[i,3])/(FTerm[i,4]-FTerm[i,3])
    else result:=0;
end;
```

Die Bestimmung der Schwerpunktskoordinate folgt den Betrachtungen von Kapitel 4. Sie ist auf drei Funktionen verteilt.

```
function TFuzzy.Termschwerpunkt(i:integer):single;
var z,n:single;
begin
    z:=(FTerm[i,2]-FTerm[i,1])*(FTerm[i,1]+2*FTerm[i,2]);
    z:=z+3*(sqr(FTerm[i,3])-sqr(FTerm[i,2]));
    z:=z+(FTerm[i,4]-FTerm[i,3])*(2*FTerm[i,3]+FTerm[i,4]);
    n:=(FTerm[i,4]+FTerm[i,3]-FTerm[i,2]-FTerm[i,1])*3;
    if n<>0 then result:=z/n else result:=FTerm[i,1];
end;


function TFuzzy.Termflaeche(i:integer):single;
begin
    result:=(FTerm[i,3]-FTerm[i,2]+FTerm[i,4]-FTerm[i,1])*FTerm[i,5]/2;
end;


function TFuzzy.GetSchwerpunktkoordinate:single;
var i:integer; A,xs:single;
begin
    xs:=0; A:=0;
    for i:=1 to FAnzahl do begin
      xs:=xs+Termflaeche(i)*Termschwerpunkt(i);
      A:=A+Termflaeche(i);
    end;
    if A<>0 then result:=xs/A else result:=Fa;
end;
```

`TFuzzy` besitzt auch einige Methoden. Eine von ihnen ist

`zeichnen`:	zeichnet die Graphen der Zugehörigkeitsfunktionen der Fuzzyvariablen (mitsamt Beschriftung)

Als Zeichenfläche dient - wie könnte es anders sein - wieder ein Objekt, hier eine Zeichenfläche vom Typ `TKoordinatensystem`. Mit der Zeile

```
EingangA.zeichnen(KS)
```

wird also das Diagramm der Fuzzyvariablen `EingangA` auf eine Zeichenfläche mit dem Namen `KS` gebracht.

Die Inferenz bezieht sich nicht nur auf eine einzige Fuzzyvariable; daher lässt sie sich nicht als Methode eines Objekts vom Typ `TFuzzy` auffassen. Weil die Inferenz auch nur an wenigen Stellen im Programm auftaucht, ist sie nicht in Form eines Objekts programmiert. In dem Array `Regel[i,j]` sind die Regeln durch Nummern erfasst: `Regel[2,3]=4` steht z.B. für: WENN EingangA = Term3 UND EingangB = Term2, DANN Ausgang = Term4. Liegt keine Regel vor, so ist `Regel[i,j]` gleich 0. Das entsprechende Programmteil ist erstaunlich kurz:

```
with Ausgang do begin
    for i:=1 to Anzahl do Term[i,5]:=0; {zuerst alle charakteristischen
                                          Werte h auf 0 setzen}
    for i:=1 to EingangA.Anzahl do for j:=1 to EingangB.Anzahl do begin
        eingang:=min(EingangA.Zugehoerigkeit[i],
                                        EingangB.Zugehoerigkeit[j]);
        if Regel[j,i]>0 then
           Term[Regel[j,i],5]:=max(eingang,Term[Regel[j,i],5]);
    end;
end;
```

Für das Pendel hingegen lohnte es sich, eine eigene Objektklasse `TPendel` zu erstellen. Sie besitzt Eigenschaften wie `Masse`, `Laenge`, `Kraft`, aber auch Methoden. Eine dieser Methoden heißt `Schritt`; sie berechnet einen Iterationsschritt gemäß Gl. (13) des Anhangs über das invertierte Pendel. Dabei fasst der Vektor *v* die Größen *x*, $\dot{x}$, α und $\dot{\alpha}$ zusammen.

```
procedure TPendel.DiffGleichung (var DGL :vektor; x :vektor);
{Dgl für invertiertes Pendel in linearer Näherung}
begin
  DGL[1]:=x[2];
  DGL[2]:=(FKraft-FcR*x[2])/Fmasse;
  DGL[3]:=x[4];
  DGL[4]:=li*(g*x[3]-(FKraft+FWind+FcR*x[2])/Fmasse-FcR*2*x[4]);
end;


procedure TPendel.Schritt;
var x_alt, DGL:vektor;
    i:integer;
begin
  for i:=1 to 4 do x_alt[i]:=v[i];
  DiffGleichung(DGL,x_alt);
  for i:=1 to 4 do v[i]:=x_alt[i]+DGL[i]*FDt;
  FZeit:=FZeit+FDt;
end;
```

Auf der CD befindet sich im Unterverzeichnis SOURCE der Quelltext eines Mondlandeprogramms. Hier können Sie an einem einfachen Beispiel lernen, wie mit den oben dargestellten Objekten bzw. Prozeduren Fuzzyanwendungen geschrieben werden können. Der Quelltext ist ausführlich kommentiert; auf ausschmückendes Beiwerk wurde bewußt verzichtet. In demselben Verzeichnis sind auch sämtliche benutzten Komponenten und Units (insbesondere die Komponente `TKoordinatensystem` und die Unit `FuzUnits`) abgespeichert. Falls Sie den DELPHI-Compiler besitzen, können Sie damit dieses Programm ausbauen oder auch andere Fuzzyanwendungen programmieren. Für den Fall, dass noch kein DELPHI-Compiler zur Verfügung steht, liegt dieses Programm auch in compilierter Form im Stammverzeichnis der CD vor. Im nächsten Abschnitt wird dieses Mondlandeprogramm

ausführlich dargestellt.

12.4 Das Mondlandeprogramm

Das Mondlandeprogramm *Mond* simuliert die Landung einer Raumkapsel auf dem Mond. Es wurde erstellt, um zu zeigen, wie man mit den im letzten Abschnitt vorgestellten Objekten bzw. Komponenten unter DELPHI in kurzer Zeit Fuzzyanwendungen programmieren kann. Von der graphischen Darstellung des Landevorgangs abgesehen sind nur ein paar Stunden für die Programmierung erforderlich. Wie Sie dem ausführlich kommentierten Quellcode entnehmen können, reduziert sich die Programmierung der Fuzzyregelung auf wenige Zeilen; der größte Teil des Quellcodes bezieht sich auf die Simulation der Bewegung der Raumkapsel sowie auf die Steuerung des Programmablaufs. Sie finden den Quellcode auf der beigefügten CD im Verzeichnis SOURCE. Sie können ihn mit einem Editor (z.B. NOTEPAD) anschauen. Zum Compilieren brauchen Sie einen DELPHI-Compiler; vor dem Compilieren müssen Sie allerdings die vom Programm benutzten Komponenten aus dem Verzeichnis SOURCE\KOMP der CD Ihrer DELPHI-Komponentenbibliothek hinzufügen.

12.5 Der Bildschirm des Mondlandeprogramms

Nicht nur in Hinblick auf die Programmierung ist dieses Mondlandeprogramm interessant; schließlich gestattet es doch, an einem anspruchsvollen Beispiel grundlegende Begriffe und Zusammenhänge

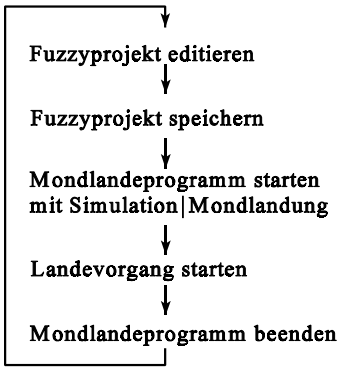
der Mechanik zu studieren und die gewonnenen Erkenntnisse zur Konstruktion einer geeigneten Fuzzyregelung zu nutzen. Aus diesem Grund liegt das Programm *Mond* auch in bereits compilierter Form vor.

Der Bildschirm des Mondlandeprogramms ist in drei Bereiche gegliedert (Abb. 12.5): Im linken Teil wird die Landung auf dem Mond graphisch dargestellt. Im rechten Bereich sehen Sie oben die Startwerte, die Sie bei Bedarf auch ändern können. Darunter befindet sich der Kontrollbereich. Hier werden die momentanen Werte für die Höhe, die Geschwindigkeit, die Zeit und den noch zur Verfügung stehenden Treibstoff angezeigt.

Im Handbetrieb kann man die Abbrandgeschwindigkeit bei den Triebwerken an dem senkrechten Feld im linken Teil des Kontrollbereichs einstellen. Dazu bewegen Sie den Mauszeiger einfach über dieses Feld. Es erscheint ein Balken, dessen Höhe ein Maß für die Abbrandgeschwindigkeit ist. Diese gibt an, wieviel Kilogramm Treibstoff pro Sekunde aus dem Raketentriebwerk ausgestoßen werden. Je größer nun diese Abbrandgeschwindigkeit ist, desto größer ist die Kraft, die die Raumkapsel abbremst oder gegebenenfalls auch nach oben steigen lässt. Die Kraft hängt auch von der Geschwindigkeit der aus dem Triebwerk strömenden Moleküle ab. Diese Geschwindigkeit ist aber durch die Zusammensetzung des Treibstoffs und den Wirkungsgrad des Triebwerks festgelegt.

Beachten Sie, dass ein Teil der Kraft des Triebwerks schon dazu aufgebracht werden muss, die Mondanziehungskraft zu kompensieren. Nur wenn die Triebwerkskraft größer ist, wird die Kapsel abgebremst. Ziel ist es natürlich, die Raumkapsel möglichst sanft, d.h. mit möglichst geringer Geschwindigkeit, auf der Mondoberfläche aufsetzen zu lassen. Zusätzlich könnte man versuchen, dies in möglichst kurzer Zeit oder mit möglichst wenig Treibstoff zu erreichen.

Um die Landung der Raumkapsel mit *Fuzzy* zu regeln, müssen Sie zunächst ein passendes Fuzzyprojekt erstellen (vgl. Aufg. 5 von Kapitel 6). Mit unserem Mondlandeprogramm können Sie zwar die Graphen der Zugehörigkeitsfunktionen anschauen, aber nicht eingeben oder verändern. Zur Eingabe des Mondlandeprojekts müssen Sie deswegen auf das Programm *Fuzzy* zurückgreifen. Die Programme *Fuzzy* und *Mond* sind datenkompatibel. Wenn Sie das Mondlandeprogramm von *Fuzzy* aus starten (dies geschieht mit dem Befehl Mond im Simulation-Menü), wird das aktuelle Projekt des *Fuzzy*-Programms automatisch an das Mondlandeprogramm übergeben; außerdem springt das Mondlandeprogramm automatisch vom Handbetrieb in den Betrieb mit Fuzzyregelung über.

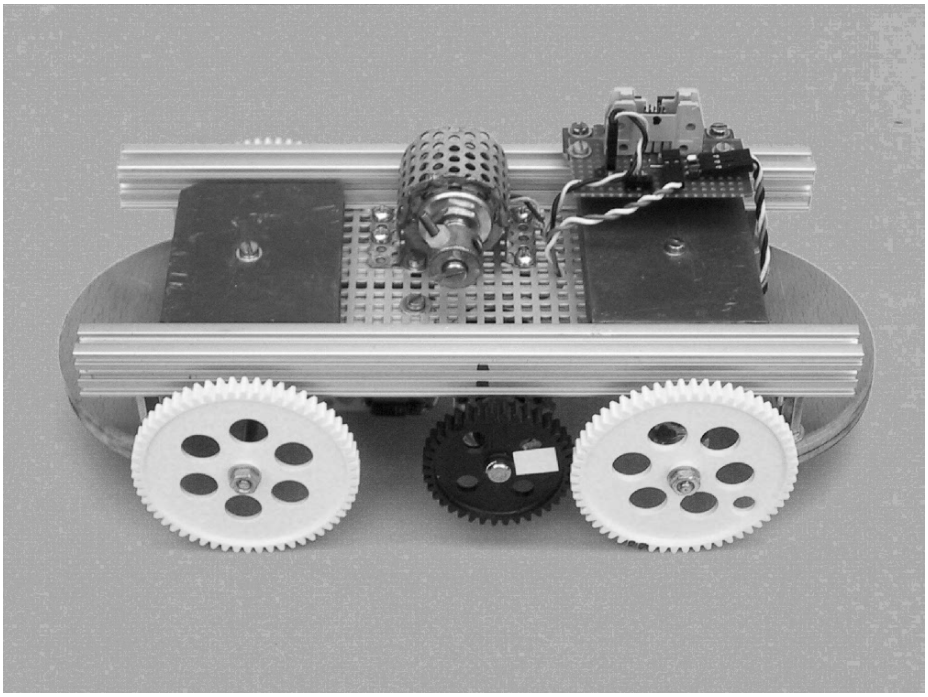


12.6 So optimieren Sie Ihr Mondlandeprojekt.

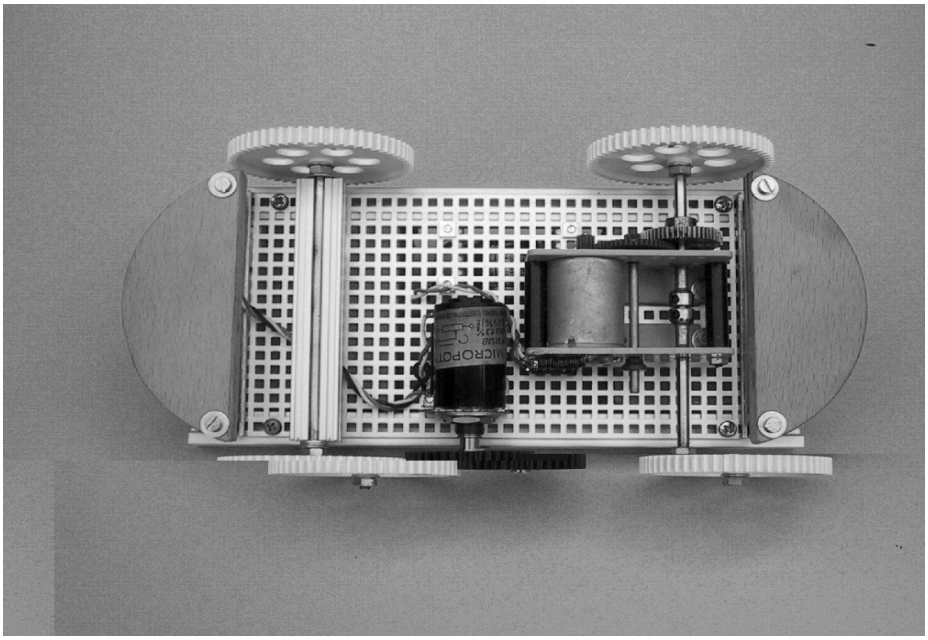
12.5 Bauanleitung für das invertierte Pendel

Der Aufbau des invertierten Pendels lässt sich in zwei Bereiche aufteilen:

- der Wagen mit Pendel, Motor und den beiden Sensoren für die Erfassung des Ablenkwinkels und der Wagenposition
- das Interface mit den beiden A/D-Wandlern, welche die von den beiden Sensoren gelieferten Messwerte dem Programm *RealExp* zuführen, sowie einer Leistungsstufe, welche den Motor ansteuert.



12.7a Wagen (von oben betrachtet). Das invertierte Pendel wird an der Schraube der Potentiometerachse (Bildmitte) befestigt.



12.7b Wagen (von unten betrachtet)

Widmen wir uns zunächst dem Aufbau des Wagens (Abb. 12.7). An einem stabilen Chassis aus Holz, Kunststoff oder Metall (Fläche etwa 10 cm mal 20 cm) werden zwei Achsen mit Rädern montiert. Diese Räder sollten beim Fahren möglichst wenig Schlupf aufweisen; wir benutzen deswegen Kunststoffzahnräder, die auf einer circa 1 m langen Schiene aus Zahnstangen fahren. Diese Schienen geben dem Wagen während des Versuchs auch eine sichere Führung.

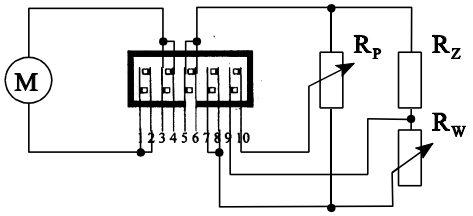
Eine der beiden Achsen wird von einem gewöhnlichen Spielzeug-Gleichstrommotor mit Getriebe (Untersetzung etwa 25:1) angetrieben[1]. Lediglich das Zahnrad, welches unmittelbar auf der Antriebsachse sitzt, sollte aus Metall sein; die restlichen Zahnräder können aus Kunststoff bestehen. An der Antriebsachse sind die Drehmomente nämlich während des Betriebs recht hoch; Kunststoffzahnräder würden dieser Belastung eventuell auf Dauer nicht Stand halten.

Zusätzlich montieren wir oberhalb des Motors einige Bleiplatten. Diese verhindern, dass der Wagen bei starker Beschleunigung über die Zähne der Zahnstangen “springt”.

Eines der Räder von der nicht angetriebenen Achse koppeln wir über ein weiteres Zahnrad an ein Zehngang-Wendelpoti (ca. 10 kΩ), welches uns an Sensor für die Position dient. Dabei sind die Verhältnisse bei den beiden Zahnrädern so gewählt, dass die Achse des Wendelpotis um 10 mal 360° gedreht wird, wenn der Wagen die gesamte Schienenstrecke abfährt.

Als nächstes montieren wir mitten auf dem Wagen ein weiteres Potentiometer. An seiner Achse befestigen wir ein Vierkant aus Holz (Maße: 1,5 cm * 1,5 cm * 100 cm). Zwei Dinge sind dabei zu beachten: Erstens muss das Potentiometer leichtgängig sein; gegebenenfalls entferne man Fett oder andere bremsende Stoffe. Zweitens muss es in sich so stabil und auch so fest auf dem Chassis befestigt sein, dass seine Achse den Belastungen durch das Holzpendel standhält. Auch hier kann man wieder ein Zehngang-Wendelpoti (10 kΩ) zurückgreifen. In diesem Fall benutzen wir einen Zusatzwiderstand R_Z (430 Ω) und stellen das Poti R_W so ein, dass sein Widerstand im Arbeitsbereich möglichst klein ist; dann sind die Spannungsänderungen pro ° am größten (Abb. 12.8).

Die Anschlüsse des Motors und unserer beiden Sensoren verbinden wir wie in der Abb. 12.8 gezeigt mit einer 10-Pol-Stiftleiste. An diese wird dann das Kabel zum Interface angeschlossen. Die Bedeutung der einzelnen Anschlüsse entnehmen Sie der Tabelle 12.1.



12.8 Schaltung auf dem Wagen

Damit ist der Aufbau des Wagens fertig. Kommen wir nun zum Bau des Interfaces. Allerdings soll an dieser Stelle nur ein Überblick gegeben werden; eine detaillierte Bestückungsanleitung, Printvorlagen für die Platinen und weitere Hinweise zum Aufbau finden Sie im Verzeichnis AUFBAU auf der CD.

Das Interface besteht aus zwei Komponenten, dem Zweifach-A/D-Wandler und der Leistungsstufe. Herz unseres Zweifach-A/D-Wandlers ist der Baustein LTC 1298[2] (Abb. 12.11). Zusammen mit den restlichen Bausteinen passt er in einen Sub-D25-Stecker, wie er üblicherweise für die Druckerschnittstelle (Centronics) benutzt wird (Abb. 12.9). Die Stromversorgung erfolgt extern über den Anschluss 3 und die Abschirmung oder über die Centronics-Schnittstelle; dazu legt die Software (*RealExp*) die

[1]Diese Artikel erhält man preiswert in Bastelläden, Baumärkten und in Elektronikversandhandel (z.B. Conrad, Völkner...)

[2]Alternativ kann der A/D-Wandler auch auf der Grundlage des Bausteins MAX 187 gebaut werden. Dieser Baustein ist der bei der Wandlung etwas schneller, dafür aber auch erheblich teurer. Eine Schaltskizze finden Sie auf der CD im Verzeichnis AUFBAU/MAX187.

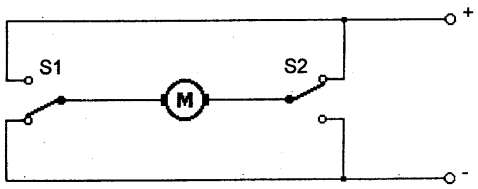
Pins 5 und 6 der Centronics-Schnittstelle auf High. Auf diese Weise kann der A/D-Wandler auch ohne weitere Stromversorgung zur Messwerterfassung eingesetzt werden.[1]

Sobald nun die Software das Kommando dazu gibt, werden die von den Sensoren gelieferten Spannungssignale vom LTC1298 gewandelt und bitweise an den Computer übergeben.

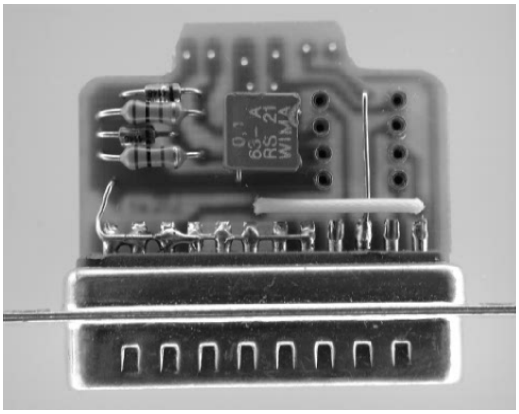
Unser A/D-Wandler besitzt noch zwei weitere Anschlüsse; hierbei handelt es sich um die beiden Leitungen zur Motoransteuerung, die hier einfach nur durchgeschleift werden. Tabelle 12.2 gibt die Bedeutung dieser beiden Motorsignale M1 und M2 an.

benutzten Impulsbreitensteuerung dargelegt worden. Bei dieser Motorsteuerung werden nur die Zustände Leerlauf, Links- und Rechtslauf benötigt; unterschiedliche Drehmomente werden durch verschiedene Impulsbreiten erzielt. Abb. 12.10 zeigt, wie die verschiedenen Zustände des Motors durch zwei Schalter S1 und S2 eingestellt werden können; allerdings befindet sich der Motor bei der Schalterstellung S1=0, S2=0 bzw. S1=1, S2=1 nicht im gewünschten Leerlauf, vielmehr wird der Motor hier durch den Kurzschluss abgebremst. Die Leistungsstufe in Abb. 11.10 sorgt dafür, dass in diesen Fällen (M1=0, M2=0) die Operationsverstärker von der elektrischen Quelle getrennt werden. Die Verbindung zwischen den Motoranschlüssen wird dann hochohmig, und die Wirbelströme sind so gering, dass sich der Motor nahezu im Leerlauf befindet.

In den beiden übrigen Fällen (M1=0, M2=1 und M1=1, M2=0) wird der Motor einmal über Op1 und einmal über Op2 mit Strom versorgt; aufgrund der entgegengesetzten Polung der Operationsverstärker läuft der Motor einmal links und einmal rechts herum.



12.10 Prinzip der Motoransteuerung



12.9 Der A/D-Wandler (IC entfernt)

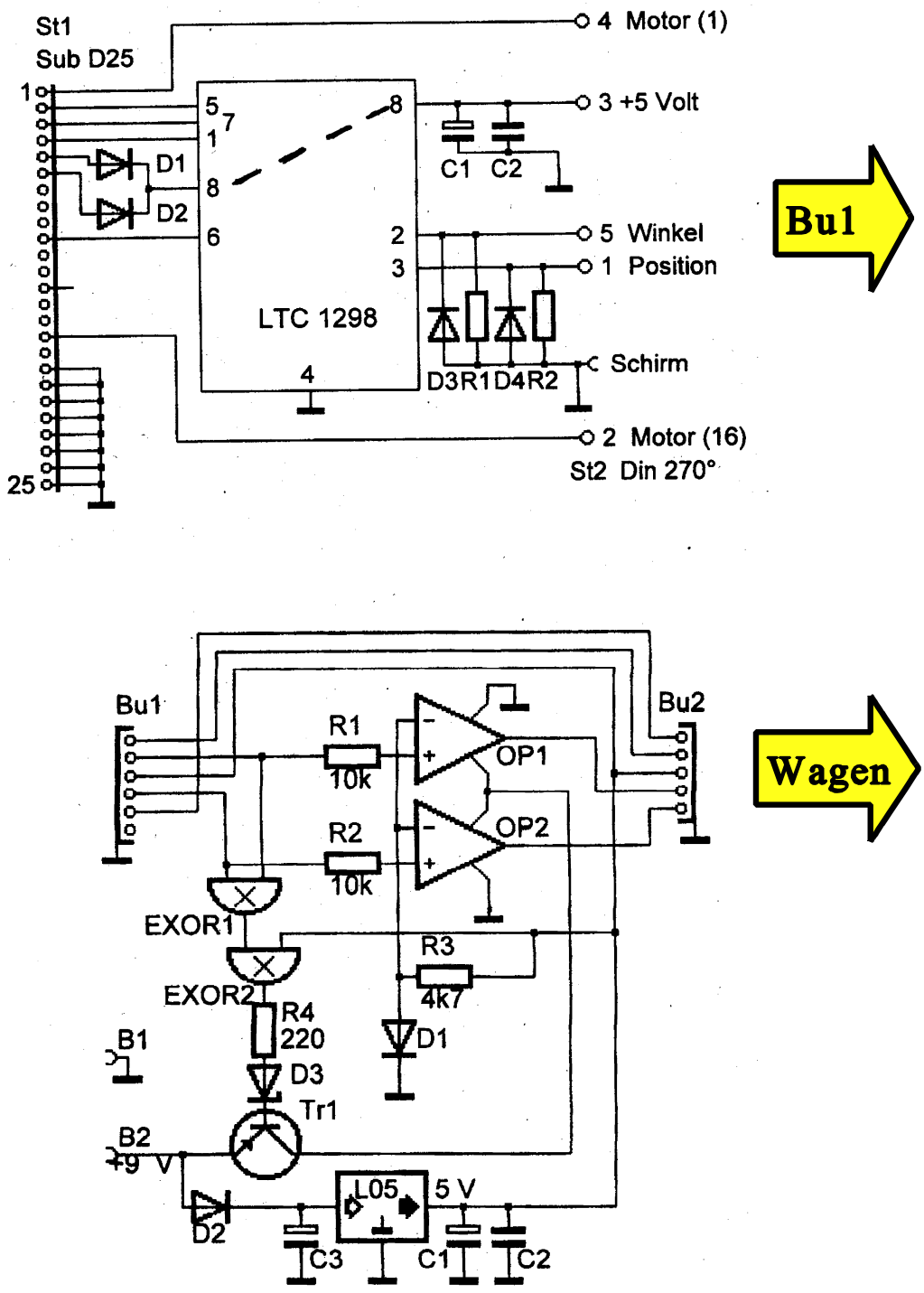
Pin-Nr. Din 180°	Stiftleiste	Funktion
1	10	Position
2	9	Winkel
3	5 u. 6	+ 5 V
4	1 u. 2	Motor
5	3 u. 4	Motor
Schirm	7 u. 8	Masse

Tab. 12.1

M1	M2	
0	0	Leerlauf
0	1	links
1	0	rechts
1	1	Leerlauf (nicht gebraucht)

Tab. 12.2

[1]Mit dem Programm *Oszi* (auf der CD im Verzeichnis Aufbau) steht Ihnen so ein preiswertes Zweikanal-Oszilloskop zur Verfügung.



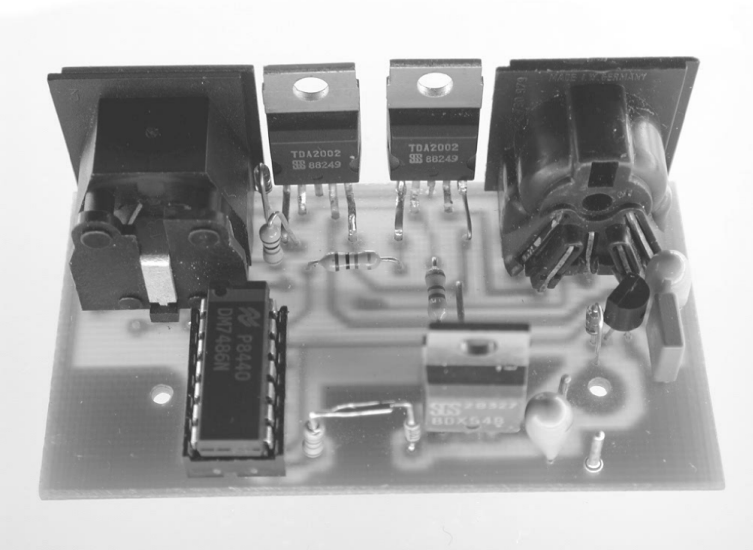
12.11 Schaltbild für A/D-Wandler (oben) und Leistungsstufe (unten)

Bezeichnung	Erläuterung	Anzahl
Bu1	DIN 6pol 270°	1
Bu2	DIN 5pol 180°	1
Bu 3, 4	Banane, isoliert	2
Platine		1
Gehäuse		1
Flachbandkabel	10pol, 2 m	1
Blechschrauben		2
D3	ZD 6V2	1
R1, R2	10 kΩ	2
R3	4,7 kΩ	1
R4	220 Ω	1
C1	4,7 µF Tantal	1
C2	100 nF MKS	1
C3	10 µF Elko	1
D1, D2	1 N 4148	2
IC1	SN 74 LS 86	1
IC2	78 L 05	1
IC3, IC4	TDA 2003	2
Fassung	14pol	1
T1	BDX 54	1
Schraube	M3 x 6	6
Mutter	M3	4
Unterlegscheibe	M3	4
Winkel	M3, Loch	2
Stiftleiste	10pol	1
Schneidklemm	10pol	1
Stecker	5pol, 180°	1

Tab. 12.3 Fuzzy-Leistungsstufe

Bezeichnung	Erläuterung	Anzahl
Platine		1
R1, R2	1 MΩ	2
C1	4,7 µF Tantal	1
C2	100 nF MKS	1
D1, D2	BAT 46	2
D3, D4	1 N 4148	2
Stecker	Sub D 25 pol	1
Gehäuse	für Sub D 25	1
Kabel	5pol geschirmt 2m	1
Stecker	5pol 270°	1
IC1	LTC 1298	1
Fassung	8pol für IC1, flach	1

Tab. 12.4 A/D-Wandler

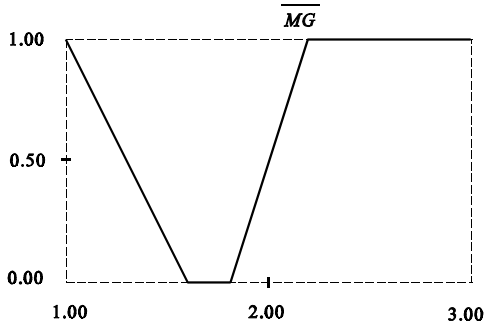


12.12 Die Leistungsstufe

12.6 Lösungen zu ausgewählten Aufgaben

2.1 Nach der Cantor'schen Betrachtungsweise gehört die Ansammlung von Sandkörnern zur Menge der Sandhaufen oder nicht. Wenn nun einzelne Sandkörner nach und nach entfernt werden, muss diese Ansammlung von Sandkörnern irgendwann wegen eines einzigen Sandkorns die Eigenschaft "Sandhaufen" verlieren. Dies ist wenig sinnvoll. Die fuzzymäßige Beschreibung bietet einen allmählichen Übergang, bei dem ein einzelnes Sandkorn nicht mehr eine alles entscheidende Rolle besitzt.

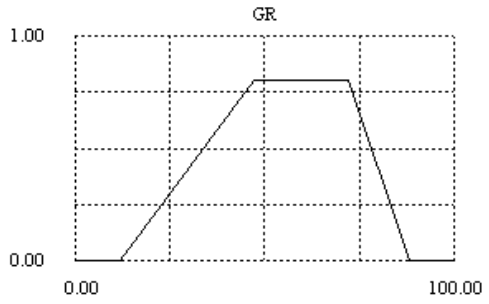
2.4



3.1

relative Luftfeuchtigkeit	Zugehörigkeitsliste Z
28%	-
45%	(1; 0)
60%	(1; 0)
70%	(0,5; 0,5)
75%	(0,25; 0,75)
80%	(0; 1)
90%	(0; 1)

3.2



3.3

Term	a	b	c	d	h
SL	-2	-2	-1	-0,5	1
LI	-1	-0,5	-0,5	0	1
NU	-0,5	0	0	0,5	1
RE	0	0,5	0,5	1	1
SR	0,5	1	2	2	1

3.4 Wir geben den Typen die Bezeichnungen a bis g gemäß der Reihenfolge in Abb. 3.10. Die Typen sind jeweils in Klammern angeben: T1 (b), T2 (a), T3 (g), T4 (c), T5 (d), T6 (e).

4.2 Die Höhe h der Graphen spielt eine Rolle bei der Bestimmung des gewichteten Mittelwerts der Schwerpunktskoordinaten der einzelnen Terme. Besteht das Ausgangsdiagramm nur aus einem einzelnen Graphen, ist die Schwerpunktskoordinate unabhängig von der charakteristischen Größe h.

4.4 $F = - 0{,}94$.

5.1 $F = 0{,}34$.

5.2 *LI* : 0,30; *NU* : 0,40; *RE* : 0,60.

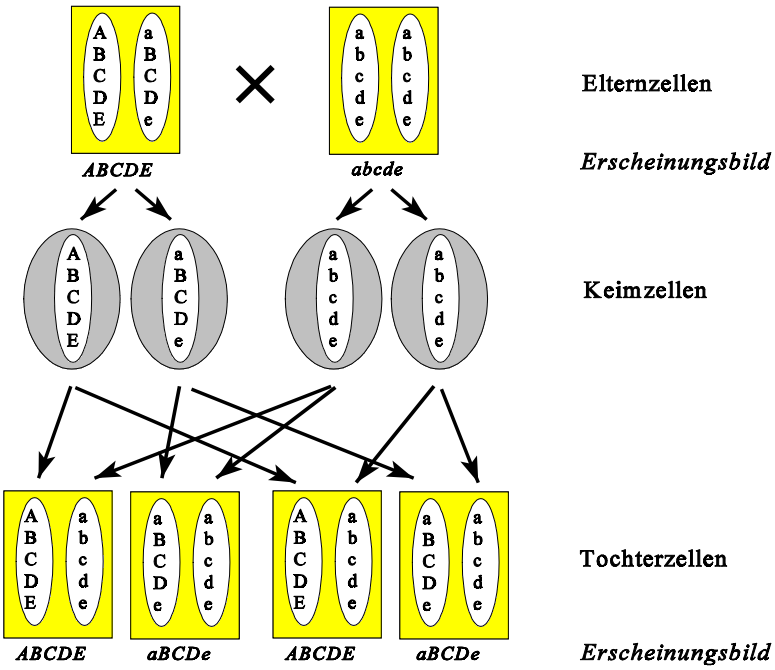
5.3 vgl. Kapitel 6.2

8.1

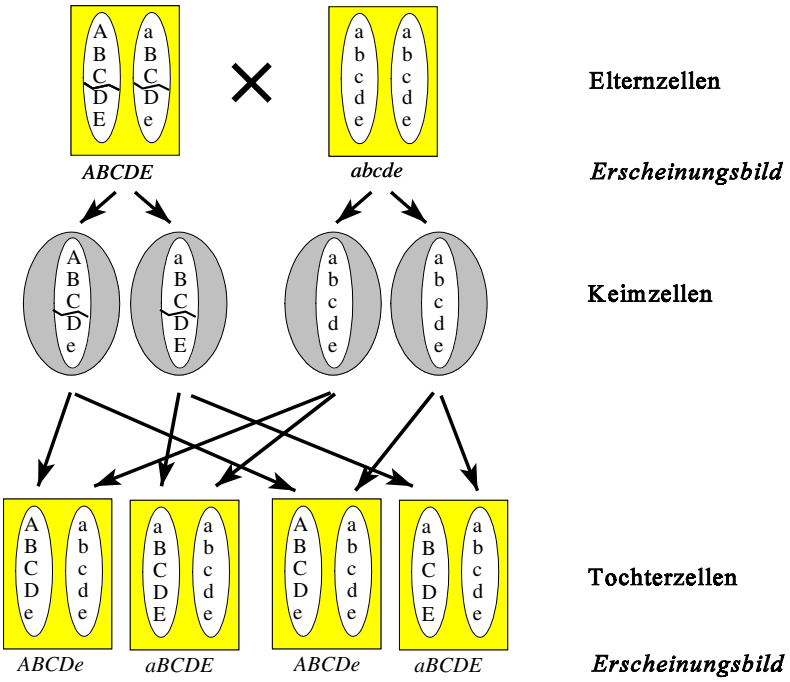
WENN Position=nah UND Winkel=links, DANN Kraft=rechts
WENN Position=nah UND Winkel=mitte, DANN Kraft=rechts
WENN Position=nah UND Winkel=rechts, DANN Kraft=rechts
WENN Position=Ziel UND Winkel=links, DANN Kraft=links
WENN Position=Ziel UND Winkel=mitte, DANN Kraft=null
WENN Position=Ziel UND Winkel=rechts, DANN Kraft=rechts
WENN Position=zu weit UND Winkel=links, DANN Kraft=links
WENN Position=zu weit UND Winkel=mitte, DANN Kraft=links
WENN Position=zu weit UND Winkel=rechts, DANN Kraft=links
WENN Position=fern UND Winkel=links, DANN Kraft=rechts
WENN Position=fern UND Winkel=mitte, DANN Kraft=stark rechts
WENN Position=fern UND Winkel=rechts, DANN Kraft=stark rechts
WENN Position=fern UND Winkel=links, DANN Kraft=rechts
WENN Position=fern UND Winkel=mitte, DANN Kraft=stark rechts
WENN Position=fern UND Winkel=rechts, DANN Kraft=stark rechts
WENN Position=nah UND Winkel=links, DANN Kraft=rechts

8.2 Ein Evolutionszyklus besteht aus der Selektion, der Reproduktion und der Mutation. Neue Chromosomen können in der Reproduktions- und in der Mutationsphase entstehen; vernichtet werden können Chromosomen in allen drei Phasen.

8.4 Die Meiose ist der Vorgang der Keimzellbildung: Fast alle Lebewesen besitzen in ihren Zellen einen doppelten Chromosomensatz. Jedes Chromosom eines Paares enthält die Erbanlagen für dieselben Merkmale, allerdings können sie sich in den "Schalterstellungen" für die Ausprägung der Merkmale unterscheiden: Sie entscheiden z.B. darüber, ob blaue oder braune Augen vorliegen. Bei voneinander abweichenden Schalterstellungen macht sich nur eine der beiden Erbanlagen im Erscheinungsbild des Individuums bemerkbar. Diese bezeichnet man als dominant, die andere als rezessiv.
Bei der Keimzellbildung werden die Chromosomenpaare aufgeteilt. Die Keimzellen, Eizellen und Spermien, enthalten nun jeweils nur einen einfachen Chromosomensatz. Bei der Befruchtung werden diese beiden Chromosomensätze neu kombiniert. Da bei den Tochterzellen nun i. a. andere Schalterstellungen kombiniert sind als bei den beiden Eltern, werden sie auch ein anderes Erscheinungsbild aufweisen. Besäßen die Organismen jeweils nur ein einziges Chromosom (in doppelter Ausführung), könnten sie höchstens vier unterschiedliche Tochterzellen erzeugen. In Abb. 12.13 ist die Kreuzung mit einem reinerbig rezessiven Partner dargestellt; die Chromosomen besitzen lediglich fünf Erbanlagen. Hier gibt es nur 2 verschiedene Tochterzellen.
Bei der Meiose kann allerdings auch ein crossing-over erfolgen: Die Chromosomen eines Paares brechen an einer Stelle auf, die einzelnen Stücke werden über Kreuz verbunden. Erst dann erfolgt die Trennung. Auf diese Weise können bei den Tochterzellen weitere Erscheinungsformen entstehen (Abb. 12.14). Das Auftauchen solcher zusätzlicher Erscheinungsformen (z.B. bei der Fruchtfliege Drosophila) ist ein Indiz für das crossing-over.



12.13 Meiose ohne crossing-over. Die dominanten Erbanlagen sind mit großen Buchstaben gekennzeichnet.



12.14 Die Meiose mit crossing-over

8.7

	Natur	Genetischer Algorithmus
1	Dieselbe Erbinformation kann auf unterschiedliche Weise codiert sein.	Codierung ist eindeutig.
2	doppelter Chromosomensatz aus mehreren Chromosomen	einfacher Chromosomensatz aus einem einzigen Chromosom
3	Veränderung bei den Tochterzellen auch ohne crossing-over oder Mutation durch unterschiedliche Kombination dominanter und rezessiver Erbanlagen	Veränderung nur durch crossing-over oder Mutation
4	Selektion erfolgt durch die Umwelt. Umwelt kann sich auf Selektionsrate auswirken.	Selektion durch Bewertungsfunktion von außen bestimmt
5	Mutation kann Chromosomenabschnitte entfernen, hinzufügen oder verändern. Wirkungen der Mutation wegen 1 und 3 sehr vielfältig	Mutation kann nur ein einzelnes Bit verändern.