ISSN 0280–5316
ISRN LUTFD2/TFRT--5462--SE

Implementering av en generell fuzzyregulator i G2

Ingemar Nilsson
Fredrik Rosberg

Institutionen för Reglerteknik
Lunds Tekniska Högskola
Oktober 1992

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
October 1992

Document Number
ISRN LUTFD2/TFRT--5462--SE

Author(s)
Ingemar Nilsson
Fredrik Rosberg

Supervisor
Karl-Erik Årzén

Sponsoring organisation

Title and subtitle
Implementering av en generell fuzzyregulator i G2 (Implementation of a general fuzzy controller in G2)

Abstract

This report describes the implementation of a general fuzzy controller in G2. The structure of the controller can be chosen freely. The rules are defined in a menu choice driven editor and the membership functions are defined in a graphical editor. Fuzzy inferencing can either be performed on-line or off-line. Off-line implies that a look-up table is generated and used for the on-line control. The operator interface shows in different colours the belief in each rule. The fuzzy value and the crisp value of the control signal can be shown in a graph.

Key words
Computer control, fuzzy controller, G2

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
Swedish

Number of pages
59

Recipient's notes

Security classification

Förord

Examensarbetet har utförts på institutionen för reglerteknik, Lunds Tekniska Högskola.

Vi vill tacka vår handledare Karl-Erik Årzén för ideér, synpunkter och värdefullt stöd.

Lund, oktober 1992

Ingemar Nilsson
Fredrik Rosberg

Innehåll

Kapitel 1

Inledning

Målet för detta examensarbete är att implementera en generell fuzzyregulator i G2. Implementeringen innefattar konfigureringsmiljö och operatörsgränssnitt. Båda dessa är grafiskt orienterade. I examensarbetet ingår också en mindre utvärdering där fuzzyregulatorn jämförs med konventionella regulatorer på en process. Beroende på tidsbrist har dettta endast utförts i begränsad omfattning.

Specifikationer:

- Både max-min och max-dot inferensmetoderna skall finnas med.
- Olika defuzzifieringssmetoder skall implementeras.
- Regulatorn skall förrutom själva den statiska fuzzydelen också innehålla möjlighet för användaren att lätt välja insignaler och utsignaler till regulatorn.
- Regulatorn skall vara generisk, dvs det skall vara lätt att skapa nya instanser av regulatorn.
- Det skall vara möjligt att bygga upp tabeller där utsignalen direkt kan avläsas för givna insignaler.
- Tillhörighetsfunktionerna skall presenteras grafiskt.

Implementeringen är gjord i G2 Version 3.0. Resultatet är en fuzzyregulator med upptill fyra in- och utgångar. Med hjälp av olika funktionsblock, t.ex. adderare och summerare, kan regulatorns inre struktur valfritt bestämmas och nya signaler, t.ex. reglerfelet, skapas av insignalerna. Dessa signaler

kopplas sedan till en statisk fuzzydel. I den statiska fuzzydelen finns automatisk generering av fuzzymängder med upptill 11 namngivna tillhörighetsfunktioner. Det är möjligt att definiera mer än 11 tillhörighetsfunktioner men då måste användaren själv namnge dem. Funktionerna definieras av 5 punktpar som kan bestämmas i en grafisk editor. Funktionerna definieras för ett normerat område som kan väljas till -1 – 1, -1 – 0 eller 0 – 1. För att anpassa insignalerna och utsignalerna till det normerade området används en skalfaktor. Skalfaktorn kan ställas under reglering. Reglerna matas in i en menystyrd editor. När fuzzyregulatorn körs kan man välja mellan att evaluera alla regler för varje sampel eller att först skapa en tabell som innehåller utsignalen för alla möjliga kombinationer av insignaler och sedan använda tabellen för att direkt bestämma utsignalerna. Att generera tabellen kan ta lång tid men tabellen möjliggör snabb reglering.

Översikt

Kapitel 2: En genomgång av fuzzyreglering. Ger en kortfattad beskrivning av teorin bakom fuzzyreglering och de olika beräkningsmetoder som används.

Kapitel 3: En beskrivning av den generella fuzzyregulator som vi har implementerat. Beskriver de olika block som finns och hur de kopplas samman. Detta kapitel utgör bruksanvisningen för programmet.

Kapitel 4: En beskrivning av implementeringen av fuzzyregulatorn i G2. Vilka datastrukturer som använts och hur olika programmeringstekniska problem har lösts. Detta kapitel förutsätter att läsaren har en grundläggande kunskap om G2.

Kapitel 5: Simulering och utvärdering av fuzzyregulatorn.

Kapitel 6: En sammanfattning av examensarbetet.

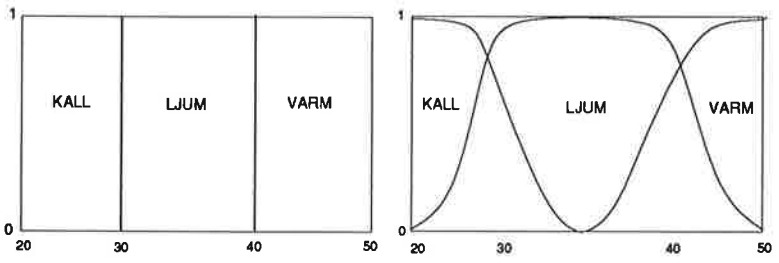
Kapitel 2

Fuzzyreglering

Fuzzy logik kan översättas till oskarp logik på svenska och används för att representera vaga uttryck som t.ex. "ökningen är stor" eller "temperaturen är hög". Detta utnyttjas i fuzzyregulatorn för att uttrycka manuella styrstrategier i form av enkla regler enligt utseendet "Om temperaturen är hög så gör en stor minskning av ångtrycket".

2.1 Fuzzymängder

Teorin om fuzzymängder introducerades av L Zadeh på 1960-talet [1] för att kunna hantera vaga och osäkra uttryck på ett matematiskt stringent sätt. I dag används fuzzylogik i bl.a. kameror, tvättmaskiner och i industrin för att reglera processer som är svåra att reglera med konventionell teknik, t.ex. roterande cementugnar [2]. En fuzzydelmängd A representeras av en tillhö-

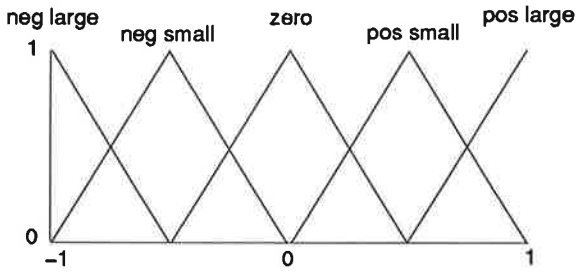


Figur 2.1: Temperaturnivåer representerade i boolsk respektive fuzzy logik

righetsfunktion $\mu_A(x)$ som för ett givet x returnerar ett värde i intervallet (0, 1). Det returnerade värdet utgör uppfyllnadsgraden eller "hur sann utsagan x tillhör A är". Till skillnad mot boolsk logik där endast värdena 0 och 1 (falsk, sann) används så kan alla värden mellan 0 och 1 (falskt, sant) antas i fuzzy logik. Fuzzy logikens mjuka gränser för en nivå, se figur 2.1, liknar en

människas sätt att uppfatta sin omgivning och detta utnyttjas i fuzzyregulatorn. I en fuzzyregulator så definierar man fuzzymängder för varje in- och utgång. Normalt så definieras fuzzymängden för ett normaliserat område te.x. -1 – 1 eller 0 – 1 och sedan används en skalfaktor för att anpassa till det verkliga området. Figur 2.2 visar en ofta använd uppsättning fuzzymängder för fuzzyregulatorer. Triangelformen är enkel att representera och det väsentliga i fuzzymängden är inte formen på tillhörighetsfunktionerna utan graden av överlappning.

För att omvandla ett exakt värde t till ett fuzzyvärde så tolkar man t som en fuzzydelmängd A med tillhörighetsfunktionen $\mu_A(x)$ där $\mu_A(x) = 0$ för alla x utom då x=t då $\mu_A(x) = 1$. A kallas för en fuzzy singelton.



Figur 2.2: Fuzzymängd för en fuzzyregulator

Tre vanliga operationer som används på fuzzymängder definieras enligt följande.

AND Snittet mellan två fuzzy delmängder A och B ges av

$$\mu_{AB}(x) = min(\mu_A(x); \mu_B(x)) \qquad \text{(vanligast)}$$

eller

$$\mu_{AB}(x) = \mu_A(x) \cdot \mu_B(x) \qquad \text{(alternativt)}$$

OR Unionen mellan två fuzzy delmängder A och B ges av

$$\mu_{A+B}(x) = max(\mu_A(x); \mu_B(x)) \qquad \text{(vanligast)}$$

eller

$$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x) \qquad \text{(alternativt)}$$

NOT Komplementet av en fuzzy delmängd A ges av

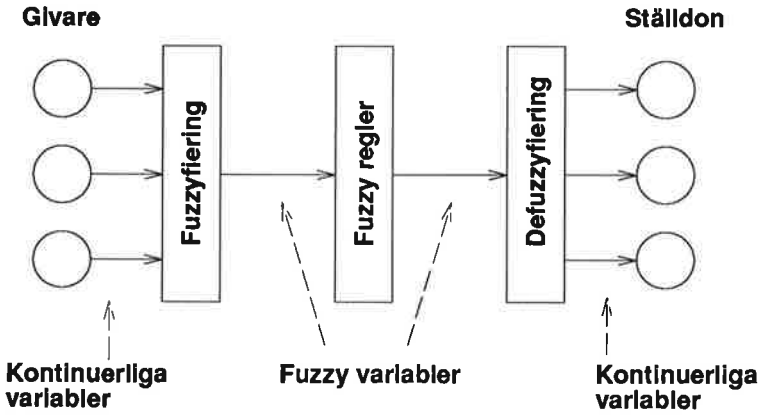
$$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$$

2.2 Fuzzyregulator

Det finns processer som är svåra att reglera med konventionella regulatorer och som därför styrs manuellt av en operatör. I en fuzzyregulator försöker man efterlikna operatörens manuella styrning av processen. Operatörens erfarenhet av processen överförs till fuzzyregulatorn i form av regler. Varje regel är en lingvistisk beskrivning av hur processen skall styras för en givet tillstånd.

```
IF tillstånd THEN åtgärd
```

Exempel på tillstånd är "temperaturen är hög" eller "trycket är mycket lågt" och exempel på några åtgärder är "öppna ventilen en aning" eller "en stor ökning av inflödet". Alla tillstånd och åtgärder är knutna till insignaler respektive styrsignaler. Varje insignal och styrsignal är uppdelad i ett antal fuzzy delmängder, där varje delmängd har en tillhörighetsfunktion och ett namn, se figur 2.2. Operatörens erfarenhet och "känsla" för processen överförs till en kunskapsbas i fuzzyregulatorn i form av regler enligt ovan.



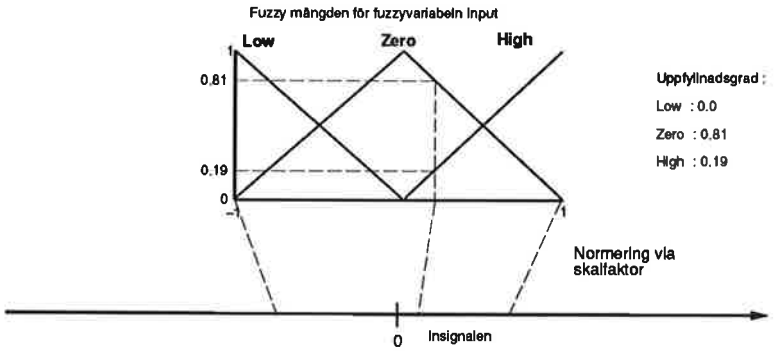
Figur 2.3: Strukturen för en fuzzyregulator

Figur 2.3 visar fuzzyregulatorns interna struktur. Insignalerna från givarna är anslutna till ett fuzzyfierings gränssnitt som skalar om insignalerna och omvandlar dem till fuzzyvärden. Inuti fuzzyregulatorn finns kunskapsbasen med fuzzyregler som avgör hur processen skall styras. Ställdonen ansluts till defuzzifieringsgränssnittet som omvandlar fuzzyregulatorns interna fuzzyvärden till exakta värden.

I fuzzyregulatorn utförs för varje sampel följande :

1. Fuzzifiering
   - Normering
   - Uppfyllnadsgraden för tillhörighetsfunktionerna beräknas
2. Fuzzy inferens, utsignalens fuzzy värde beräknas
3. Defuzzifiering
   - Utsignalens exakta värde beräknas
   - Den normerade utsignalen skalas om

Fuzzyfiering

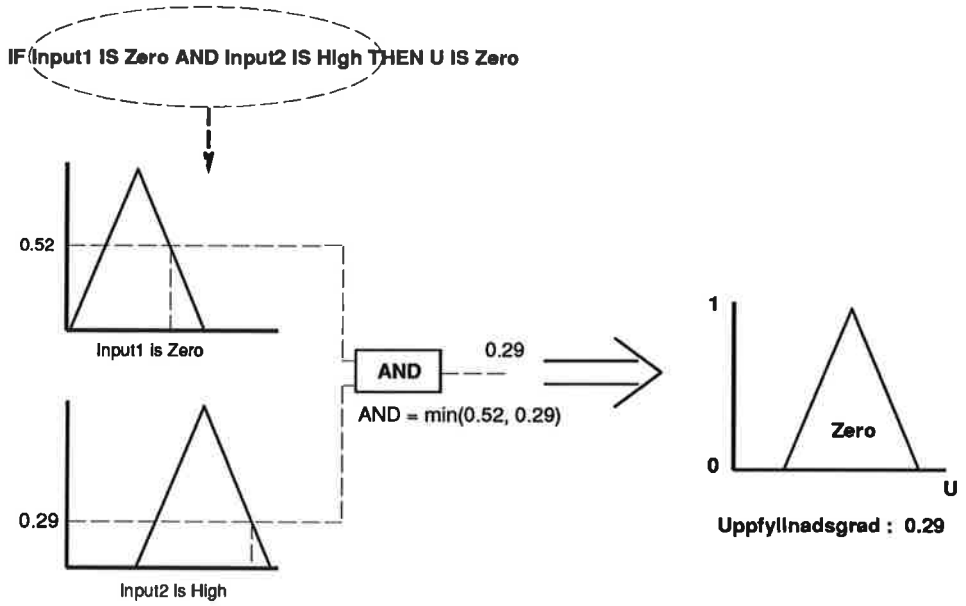


Figur 2.4: Fuzzyfiering

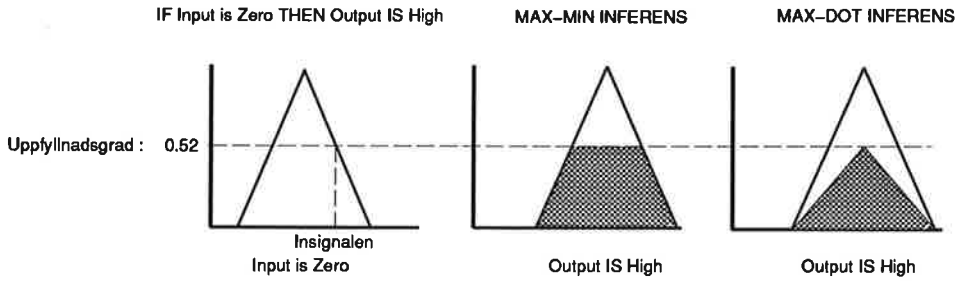
Under fuzzifieringen så normeras insignalerna och därefter bestäms till vilken grad som tillstånden i reglerna är uppfyllda, se figur 2.4. Uppfyllnadsgraden för varje regel beräknas utifrån detta. Uppfyllnadsgraden, "hur sann en regel är", är ett värde i intervallet (0, 1). Normalt så har flera regler samtidigt en uppfyllnadsgrad som är större än 0. Figur 2.5 visar hur uppfyllnadsgraden för en regel beräknas.

Inferens

Efter fuzzifiering följer inferens där reglernas styråtgärder viktas efter reglernas uppfyllnadsgrad och styråtgärder som verkar på samma styrutgång kombineras samman. Det finns olika metoder att vikta styråtgärder. De två metoder som är implementerade i fuzzyregulatorn är max-min och max-dot metoden. Max-min metoden beräknar styråtgärdens tillhörighetsfunktion μ_w som $\mu_w = min(b, \mu_x)$ och med max-dot fås $\mu_w = b \cdot \mu_x$ där b är uppfyllnadsgraden för regeln och μ_x är tillhörighetsfunktionen för styråtgärden. Se



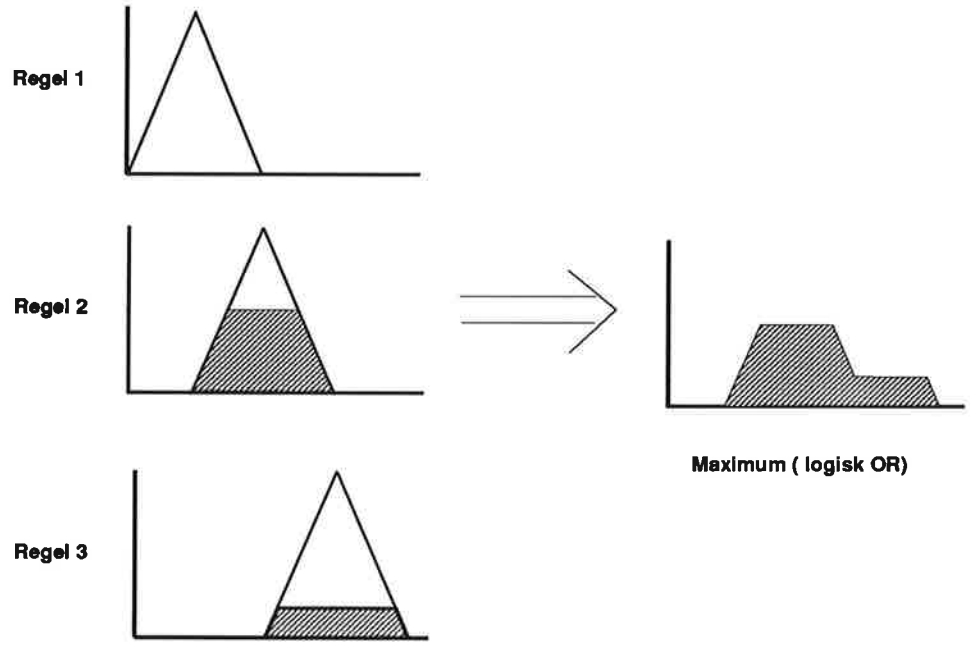
Figur 2.5: Fuzzyfiering



Figur 2.6: Max-min och max-dot metoden

figur 2.6. Max-dot metoden bevarar tillhörighetsfunktionens utseende till skillnad ifrån max-min metoden.

När varje regels bidrag till styråtgärdena har beräknats enligt ovan så skall de kombineras samman genom att använda operationen OR (max). Styråtgärder som verkar på samma utgång kombineras samman. Resultatet blir utsignalens fuzzyvärde, se figur 2.7.



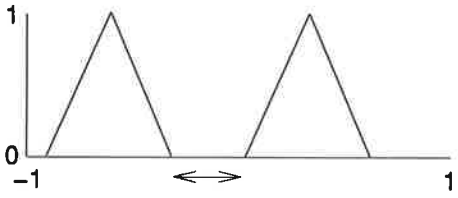
Figur 2.7: Styråtgärder som verkar på samma utgång kombineras samman

Defuzzifiering

Den beräknade styrsignalen utgörs av ett fuzzyvärde som representeras av en tillhörighetsfunktion μ_x. Under defuzzifiering omvandlas fuzzyvärdet till ett exakt värde som sedan skalas om och blir regulatorns utsignal. Det finns ett flertal olika defuzzyfieringsmetoder.

MOM (Mean of Max): Välj det x som maximerar μ_x. Om det existerar fler maxima används medelvärdet av maxima. Denna metod ger inget kontinuerligt samband mellan fuzzyregulatorns insignaler och utsignaler.

COA (Center of Area): Välj det x som delar arean under μ_x i två lika stora bitar. Denna metod är inte alltid väldefinierad, se figur 2.8.

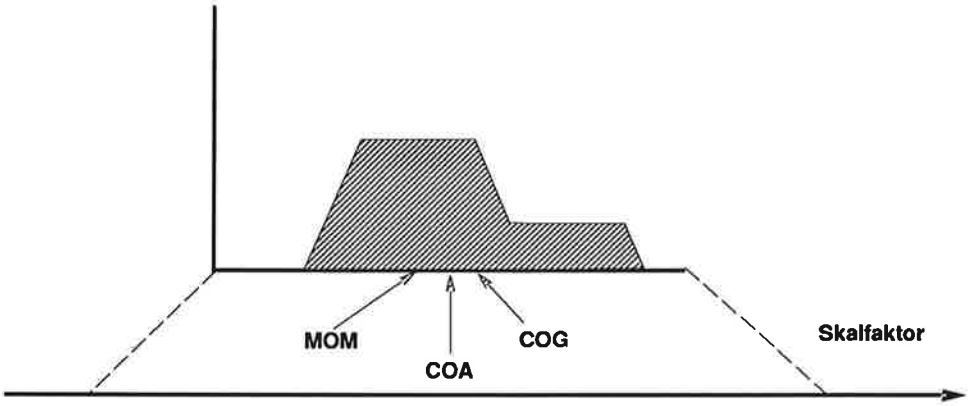


Figur 2.8: Odefinierat tillstånd för COA

COG (Center of Gravity): Beräkna det x som utgör tyngdpunkten för $\mu(x)$. Detta är den vanligaste av de uppräknade metoderna.

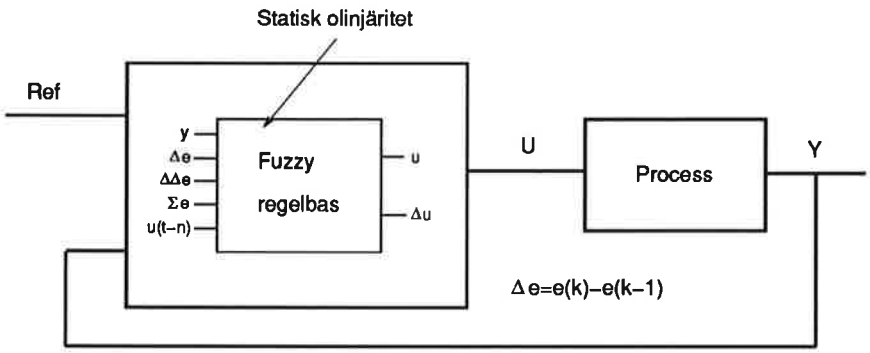
$$x = \frac{\int x\mu(x)}{\int \mu(x)}$$

Figur 2.9 visar den resulterande utsignalen för de olika defuzzifieringsmetoderna.



Figur 2.9: Olika defuzzifieringsmetoder

2.2.1 Olika strukturer för en fuzzyregulator

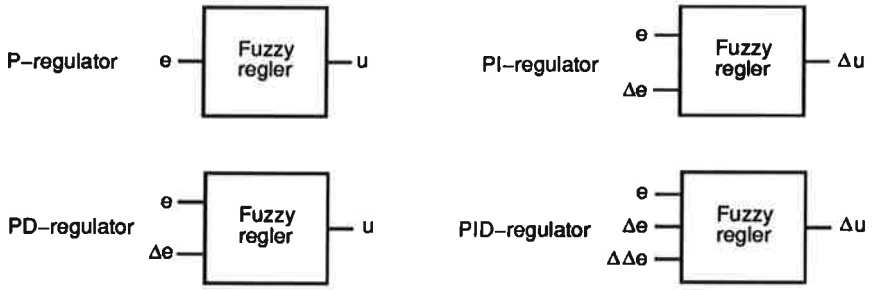


Figur 2.10: Regulator struktur

En fuzzyregulator är en olinjär regulator. Hur olinjäriteten ser ut beror på vilka beräkningsmetoder som används i regulatorn, hur man har valt tillhörighetsfunktionerna för fuzzyvariablerna samt vilka regler som används. Normalt så finns det ett statiskt samband mellan insignalerna och utsignalerna i en fuzzyregulator. En fuzzyregulator kan därför betraktas som en stor tabell där insignalerna pekar ut utsignalerna i tabellen.

Till fuzzyregulatorn ansluts olika signaler, se figur 2.10. Exempel på lämpliga signaler är reglerfelet, första differensen av reglerfelet, andra differensen

av reglerfelet etc. Genom att använda dessa signaler kan man skapa fuzzy-regulatorer med olika karaktär, se figur 2.11.



Figur 2.11: Några fuzzyregulator strukturer

2.2.2 Fördelar och nackdelar med fuzzyreglering

Fördelen med fuzzyreglering är att man slipper att identifiera och göra en matematisk modell av processen som skall styras. Istället kan man direkt automatisera en manuell styrstrategi. Detta är speciellt fördelaktigt då det är svårt att skapa en modell av processen och därför svårt att styra processen med konventionella linjära metoder [3] [4].

En nackdel med fuzzyregulatorer är att det inte går att garantera att systemet är stabilt. Det finns inte heller någon systematisk metod för att välja regler till regulatorn. Istället får man utnyttja sin erfarenhet och kunskap om processen som skall regleras för att formulera fuzzyregler och sedan pröva sig fram för att optimera regulatorn.

2.2.3 Implementering av fuzzyreglering

Hårdvara

Den första integrerande kretsen för fuzzy logik kom 1985 och hade utvecklats av Togai och Watanabe vid AT&T Bell laboritories. Kretsen klarar 250 000 FLIPS (Fuzzy Logic Inferences Per Second). Det finns idag fuzzy-datorer som klarar mer än 10 mega-FLIPS.

CAD, mjukvara

Det finns kommersiella CAD paket för att utveckla fuzzyregulatorer. Dessa innehåller ett konfigureringsgränssnitt för att konstruera fuzzyregulatorn. Programmet genererar sedan källkoden i ett högnivå programmeringsspråk, t.ex. C, som sedan kan användas i en konventionell dator. Alternativt så genererar CAD-paketet kod för speciell fuzzy hårdvara som implementerar fuzzy regleralgoritmen direkt i hårdvaran.

Mjukvara

Det är förhållandevis enkelt att implementera en fuzzyregulator. Men ett program som utför de olika stegen i fuzzyalgoritmen som beskrivits tidigare kräver att många beräkningar måste utföras för varje sampel.

Ett sätt att snabba upp fuzzyregulatorn är att skapa en tabell (lookup table) som innehåller utsignalen för alla möjliga kombinationer av insignaler. Denna tabell skapas off-line och vid reglering så ger tabellen utsignalen direkt för en given insignal. Antalet kvantiseringssteg för insignalerna i tabellen innebär en avvägning mellan hur noggrann regleringen måste vara och hur mycket minnesutrymme som kan upplåtas för tabellen.

Att använda en tabell innebär en diskretisering av utsignalen. Det betyder att det kan uppstå ett kvarstående fel och att regulatorn står och svänger mellan närliggande värden i tabellen, se exempel i kapitlet . För att förbättra tabellen kan man interpolera mellan tabellens värden. Interpolering kräver omfattande beräkningar om fuzzyregulatorn har mer än två ingångar. I vår fuzzyregulator sker ingen interpolering.

Ett sätt att använda tabellslagning på, är att designa fuzzyregulatorn på en kraftfull dator typ arbetsstation, skapa tabellen och sedan ladda ner tabellen i den egentliga reglerdatorn. På så sätt kan även en mindre dator köra fuzzyreglering.

Kapitel 3

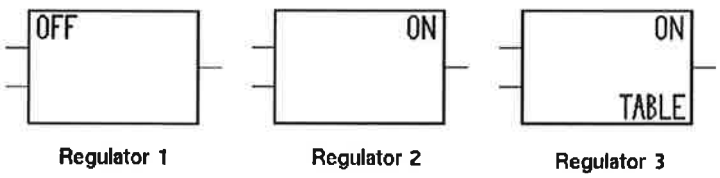
Funktionsbeskrivning

Detta kapitel ger en allmän beskrivning av den fuzzyregulator vi har implementerat i G2. Kapitlet utgör en bruksanvisning och beskriver hur en fuzzyregulator byggs upp i det grafiska konfigureringsgränssnittet.

3.1 Fuzzyregulatorn

För att de olika funktionerna i fuzzyregulatorn skall fungera måste G2:s **user mode** vara **user**.

En färdig regulator består av ett block med ett antal ingångar och utgångar. Beroende på om regulatorn körs eller är frånslagen så är blockets utseende olika.



Figur 3.1: Regulatorblock i olika moder

Figur 3.1 visar regulatorblock i tre olika moder. Regulator 1 är frånslagen, regulator 2 körs, regulator 3 körs med tabellslagning. I menyn som kommer upp när man klickar på regulatorblocket finns menyval för att stoppa regleringen : **control off**, starta regleringen : **control on** och starta reglering med tabellslagning : **control on table**. Regleringen går endast att starta om alla fuzzyvariabler är initierade. Om någon fuzzyvariabel inte är initierad meddelar programmet vilken variabel det är. Initieringen beskrivs

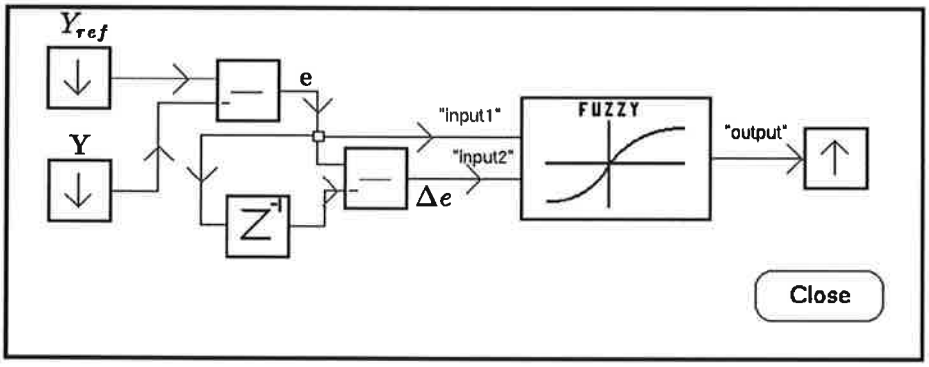
längre fram i kapitlet.

Att starta reglering med tabellslagning innebär att en tabell för regulatorn skall skapas. En ny tabell skapas endast om regulatorn har ändrats sen senaste körningen. Under reglering så pekar regulatorns insignaler på utsignalen i tabellen. Detta medför att exekveringen går snabbare än om alla regler skall evalueras för varje sampel.

Att skapa en tabell tar lång tid, från ett par minuter för en liten regulator med två ingångar till timmar för större regulatorer. För att kunna avbryta tabelluppbyggnaden så finns det en knapp för att avbryta: **abort**. Dessutom visas hur stor del av tabellen som har skapats (fungerar endast om G2 körs i realtidsmode).

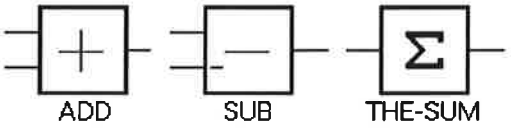
3.2 Regulatorns byggblock

Genom att välja **go to subworkspace** i menyn till ett regulatorblock så öppnas regulatorn. På det fönster som öppnas finns de block som bygger upp regulatorn samt en knapp för att stänga fönstret: **close**.



Figur 3.2: Ett öppnat regulatorblock

Figur 3.2 visar en fuzzyregulator med två ingångar och en utgång. Med en subtraherare skapas reglerfelet e(k) som skillnaden mellan de båda ingångarna och differensen mellan nuvarande och föregående fel skapas med ytterligare en subtraherare och ett fördröjningsblock. Nedan följer en beskrivning av de funktionsblock som är tillgängliga. Det är även möjligt att skapa egna block, detta beskrivs i kapitlet Implementering.



Figur 3.3: Adderare, Subtraherare och Summerare

Additionsblocket, figur 3.3, summerar de två insignalerna enligt;

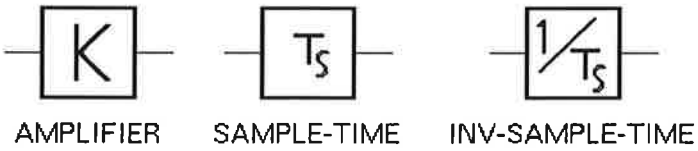
$$U_{ut}(t) = U_{in1}(t) + U_{in2}(t)$$

Subtraheringsblocket, figur 3.3, subtraherar insignalen på ingången med litet minustecken ifrån insignalen på den andra ingången.

$$U_{ut}(t) = U_{in}(t) - U_{in-}(t)$$

Summationsblocket, figur 3.3, summerar insignalen med den tidigare utsignalen. För att sätta en begränsning på totala summan i summationsblocket klicka på menyvalet **set sum limit**. Begränsningen gäller i både det positiva och negativa området. En inmatningsruta kommer att dyka upp. Det inmatade värdet kan vara antingen heltal eller flyttal. När summatorn kommer upp i till sin gräns stannar summationen åt det håll som skulle innebära att begränsningen passerades. Formeln för summationsblocket blir alltså;

$$U_{ut}(t) = U_{ut}(t-1) + U_{in}(t) \qquad \text{[-begränsning, +begränsning]}$$



Figur 3.4: Förstärkningsblock och tidskonstanter

Förstärkningsblocket, figur 3.4, multiplicerar insignalen med K, där K bestäms av användaren. För att sätta K i förstärkningsblocket klicka på menyvalet **set k**. En inmatningsruta kommer att dyka upp. Det inmatade värdet kan vara antingen heltal eller flyttal. Formeln för förstärkningsblocket är :

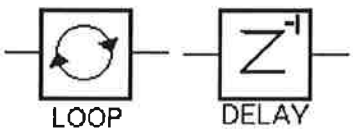
$$U_{ut}(t) = K \cdot U_{in}(t)$$

Tidskonstantblocket, figur 3.4, multiplicerar resp. dividerar insignalen med regulatorns samplingstid h. Samplingstiden för regulatorn sätts i det översta blocket, där man även startar och stoppar regleringen.

$$U_{ut}(t) = U_{in1}(t) \cdot h$$

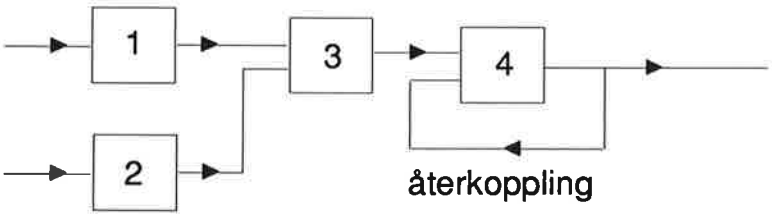
resp.

$$U_{ut}(t) = U_{in1}(t)/h$$



Figur 3.5: Loop- och Fördröjningsblock

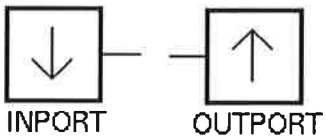
Loopblocket, figur 3.5, används när man gör en återkoppling i nätet. Om man inte använder loopblocket för att göra en återkoppling, från en utgång i nätet till en ingång tidigare i nätet, kommer fel uppstå vid försök att köra regulatorn.
Blocken exekveras ett taget och i ordning så att signalen propagerar från ingång till utgång på regulatorn. I figur 3.6 innebär detta att block 1 och 2 måste exekveras före block 3, annars är inte ingångarna på block 3 definierade. I block 4 har man gjort en återkoppling och det innebär att block 4 skall exekveras före block 4 för att ingångarna skall vara definierade: en motsägelse. Med ett loopblock i återkopplingsslingan löses detta problem. Genom att låta loopblocken exekveras före de andra blocken kommer det blocket som är anslutet till utgången på loopblocket att ha en definierad insignal.



Figur 3.6: Algebraisk loop

Fördröjningsblocket, figur 3.5, fördröjer signalen en tidsenhet.

$$U_{ut}(t) = U_{in}(t-1)$$

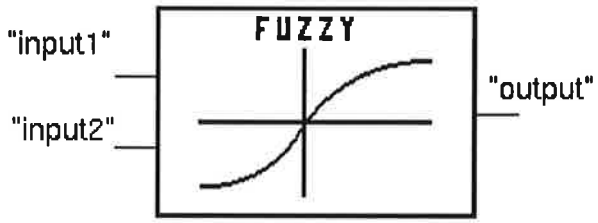


Figur 3.7: Regulatorns in och utport

Inportar, figur 3.7, är de block som är knutna till respektive ingång på regulatorblocket.

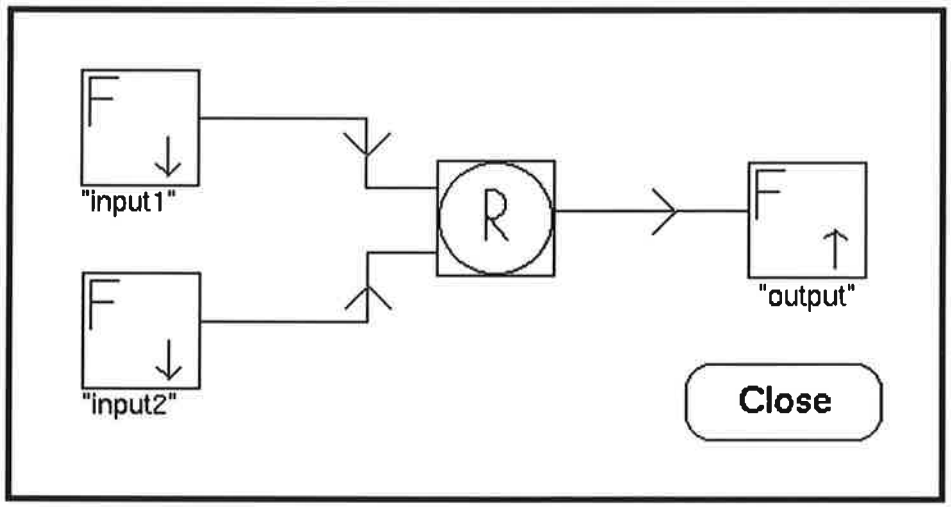
Utportar, figur 3.7, är de block som är knutna till respektive utgång på regulatorblocket.

Man får inte dubblera in- och utportar, ta bort in- och utportar, osv. eftersom dessa handlingar kommer att leda till fel i regulatorn.



Figur 3.8: Ett statiskt fuzzyblock

Figur 3.8 visar blocket som innehåller den statiska fuzzydelen i regulatorn. Genom att öppna blocket får man tillgång till fuzzyvariablerna och fuzzyreglerna.



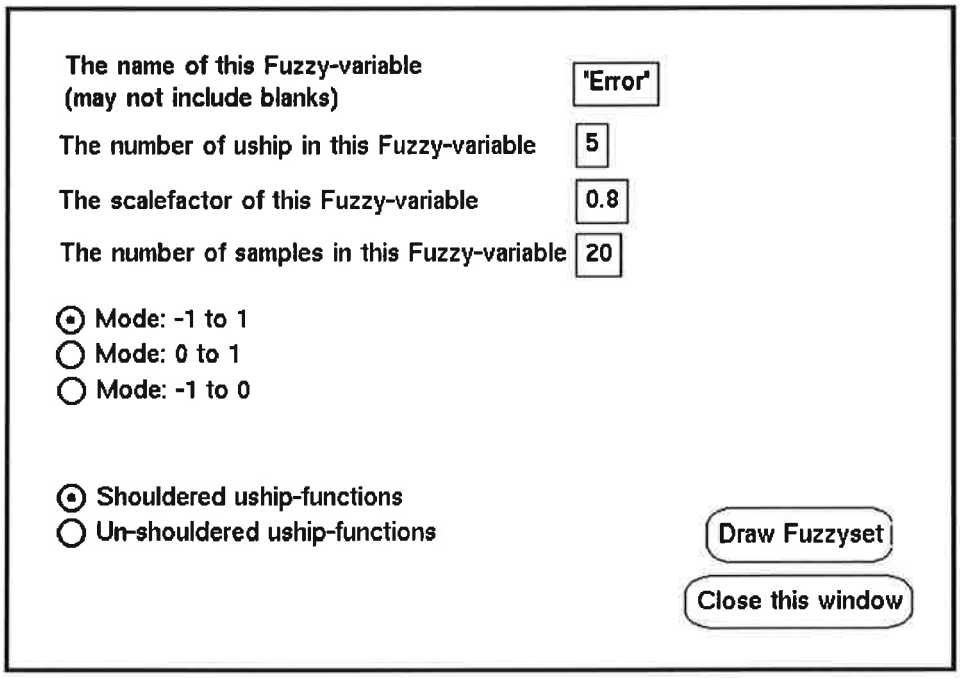
Figur 3.9: Ett öppnat statiskt fuzzyblock

Figur 3.9 visar ett öppnat statiskt fuzzyblock. Blocken märkta med F är fuzzyvariabler och motsvarar anslutningarna på det statiska fuzzyblocket. Genom att öppna en fuzzyvariabel får man upp ett fönster där man kan bestämma fuzzyvariabelns tillhörighetsfunktioner, namn etc. Detta beskrivs nedan. Blocket märkt med R är regelblocket och innehåller en regeleditor som kommer fram när blocket öppnas. Innehållet i det statiska fuzzyblocket är förutbestämt och får inte ändras av användaren.

3.3 Fuzzyvariablerna

Vilka parametrar man kan ändra på och vad som händer när man öppnar en fuzzy-variabel beror på om regleringen körs eller frånslagen.

3.3.1 Regulatorn frånslagen



Figur 3.10: En öppnad fuzzyvariabel när regulatorn är frånslagen

När man väljer **open** i menyvalet till en fuzzy-variabel kommer man att få upp ett fönster enligt figur 3.10. När text eller siffror skall matas in trycker man på inmatningsrutan. Endast en fuzzy-variabel kan öppnas i taget.

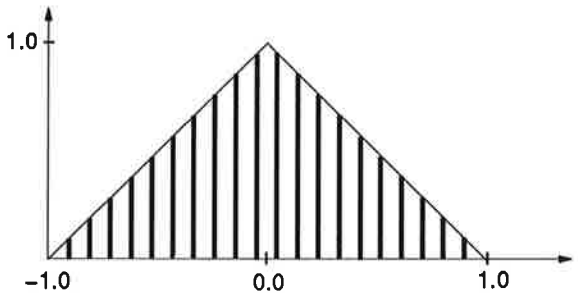
The name of this Fuzzy-variabel: I denna inmatningsruta skriver man namnet på sin fuzzyvariabel. Namnet kommer att visas dels under fuzzy-variabeln men även vid den in- eller utgång på det statiska fuzzyblocket som är knuten till fuzzy-variabeln. Observera att namnet måste börja med ett (") och även avslutas med (") samt att namnet inte får innehålla mellanslag.

The number of uship in this Fuzzy-variable: Här anger man önskat antal tillhörighetfunktioner. Om ifyllt antal är mindre än tre kommer programmet att initiera fuzzy-variabeln med tre tillhörighetsfunktioner.

Om man vill ha mindre än tre tillhörighetsfunktioner går man in i den grafiska editeringsmiljön till fuzzy-variabeln, se sektion 3.3.3 , och tar bort önskat antal tillhörighetsfunktioner.
Det finns ingen övre gräns på antalet tillhörighetsfunktioner som kan initieras . Programmet kommer att initiera namn på alla tillhörighetsfunktioner om antalet är högst elva. Om inmatat antal är högre än elva måste användaren själv namnge varje tillhörighetsfunktion.

The scalefactor of this Fuzzy-variabel: Inmatning av skalfaktorn som anpassar det normaliserade området (enligt **Mode** nedan) till verkliga signalnivåer. Skalfaktorn kan även användas som en förstärkningsparameter i regulatorn.

The number of samples in this Fuzzy-variable: Anger antalet kvantiseringsnivåer som varje tillhörighetsfunktioner skall ha. Vid körning av programmet kvantiseras tillhörighetsfunktionerna enligt figur 3.11.

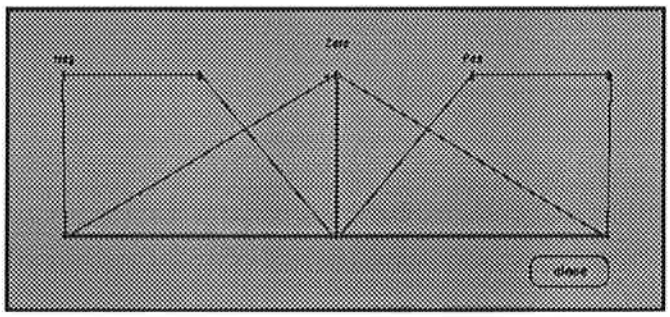


Figur 3.11: Kvantisering, i 22 kvantiseringsnivåer varav en är placerad i -1.0 och en annan i 1.0, av en tillhörighetsfunktion

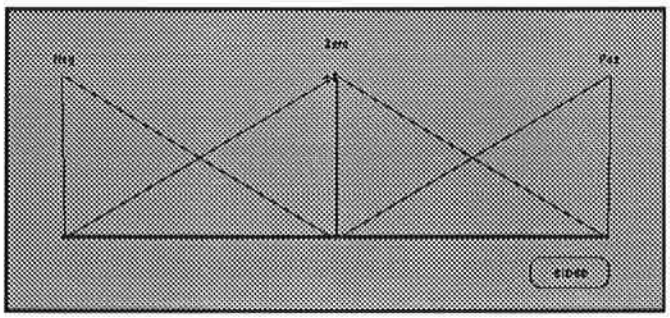
I programmet kvantiseras (*samplar*) tillhörighetsfunktionerna och de kvantiserade värdena sparas i en vektor.
I figur 3.11 är det de kraftiga linjerna som representerar de kvantiserade värdena.

Med ett högt värde får man en hög upplösning medan ett lågt värde ger en dålig upplösning. Tänk dock på att vid körning av regulatorn med tabellslagning är tiden för uppbyggnaden av tabellen beroende av antalet kvantiseringnivåer.

Mode: Önskat normaliseringsområde väljs genom att klicka på knapparna. Valet av område bestäms av de signaler som fuzzy-variabeln skall representera. Med signaler som är både positiva och negativa väljer man lämpligen området -1 till 1. Om man däremot har endast negativa eller endast positiva signaler blir upplösning bättre om man väljer -1 till 0 för negativa signaler och 0 till 1 för positiva signaler.



Figur 3.12: Fuzzyvariabler med avfasning



Figur 3.13: Fuzzyvariabler utan avfasning

Shouldered or unshouldered uship-functions: Här väljer man om tillhörighetsfunktionerna skall ha avfasning, figur 3.12, eller vara utan avfasning, figur 3.13.

Draw Fuzzyset: Fuzzyvariabeln ritas ut i en grafisk editeringmiljö, se 3.3.3.

Close this window: Stänger fönstret när man är nöjd med de inmatade värdena.

Observera att det är först när man trycker på knapparna "Stäng fuzzyvariabeln" eller "Öppna den grafiska editeringsmiljön" som ändringar i inmatade värden genomförs.

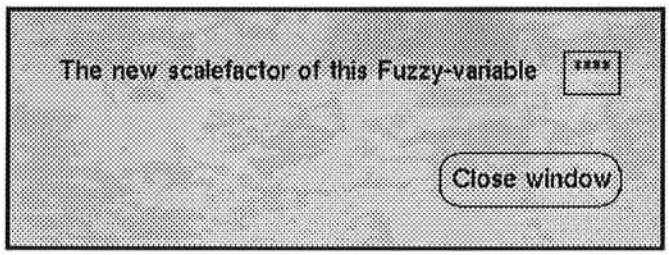
3.3.2 Regulatorn körs

När regulatorn kör och man trycker på en fuzzy-variabel kommer det upp nya menyval för fuzzyvariablerna.

Show uhsip function: Vid detta val kommer fuzzyvariablernas tillhörighetsfunktioner att visas grafiskt. Med tre tillhörighetsfunktioner blir

den grafiska presentationen enligt figur 3.12 med avfasning eller figur 3.13 utan avfasning.
Observera att ingen editering kan ske i denna miljö.

Show output graph: Detta val finns endast för utvariabler. Om man väljer detta menyalternativ visas ett diagram som presenterar utsignalens fuzzyvärde samt utsignalens exakta värde iform av en pil i diagrammets underkant. Klickar man på diagrammet så försvinner det.



Figur 3.14: En öppnad fuzzyvariabel när regulatorn kör

Open: Det enda värde som kan ändras när regulatorn kör är skalfaktorn. Vid detta menyval får man upp ett fönster med en inmatningsruta för ändring av skalfaktorn, se figur 3.14.

3.3.3 Den grafiska editerings miljön

Vid valet **Draw Fuzzyset** i den öppnade fuzzyvariabeln kommer det att dyka upp ett fönster enligt figur 3.15.
Endast en fuzzyvariabel i taget kan editeras grafiskt.

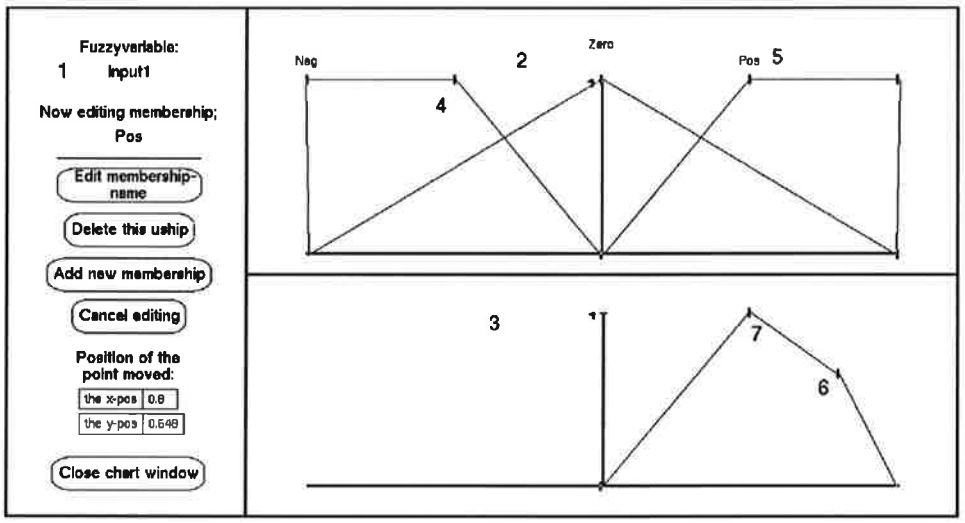
Siffrorna nedan relaterar till siffrorna i figur 3.15.

Fönster nummer 1: Alla knappar i detta fönster gäller för tillhörighetsfunktionen som finns i editeringsfönstret (nummer 3).

Fuzzyvariable: Här anges namnet på den fuzzyvariabel vars tillhörighetsfunktioner editeras i dessa fönster. I figur 3.15 är det tillhörighetsfunktionerna till fuzzyvariabeln **Input1** som skall editeras.

Now editing membership: Här anges namnet på den tillhörighetsfunktion som editeras (finns i fönster nummer 3) för tillfället. I figur 3.15 är det **Pos** som editeras.

Edit membershipname: Namnet på den tillhörighetfunktion som befinner sig i editeringsfönstret ändras till det som man skriver i inmatningsrutan som dyker upp.



Figur 3.15: Den grafiska editerings miljön med en tillhörighetsfunktion i editeringsfönstret (nederst)

Delete this uship: Tar bort tillhörighetsfunktionen, som finns i editeringsfönstret, ifrån fuzzyvariabeln.

Add new membership: En tillhörighetfunktion med slumpmässig placering kommer att adderas till fuzzyvariabeln. En inmatningsruta, där man skall skriva in namnet på den nya tillhörighetsfunktionen, kommer att dyka upp. Efter det att ett namn är ifyllt kommer den nya tillhörighetfunktionen att direkt placeras i editeringsfönstret.

Cancel editing: Denna knapp avbryter editeringen av den tillhörighetfunktion som befinner sig i editeringsfönstret, fönster nummer 3. Tillhörighetfunktionen kommer att behålla sitt gamla utseende (den kommer att se ut som den gör i fönster nummer 2). För att stänga av den grafiska editerings miljön se nedan.

Position of the point moved: Här visas koordinaterna på den senast flyttade punkten angivet i x- och y-led. I bild 3.15 är det koordinaterna för punkten vid siffran 6 som anges eftersom det är denna punkt som flyttades senast.

Close chart window: Med denna knapp stänger man den grafiska editerings miljön. Om det finns någon tillhörighetsfunktion i fönster nummer 3 kommer ändringarna av den tillhörighetsfunktionen att sparas.

Fönster nummer 2: Fönster 2 visar fuzzy-variabelns tillhörighetsfunktioner. När en tillhörighetfunktion finns i editeringsfönstret kommer den

byta färg i fönster 2 (för att man skall kunna urskilja vilken tillhörighetfunktion det är som editeras).

Punkter i Tillhörighetsfunktion: Vid siffra 4 i figur 3.15. Varje tillhörighetsfunktion är grafiskt representerade av ett antal punkter, sammankopplade med dragna linjer. Om man trycker på en punkt i detta fönster kommer tillhörighetsfunktionen att överföras till editeringsfönstret.

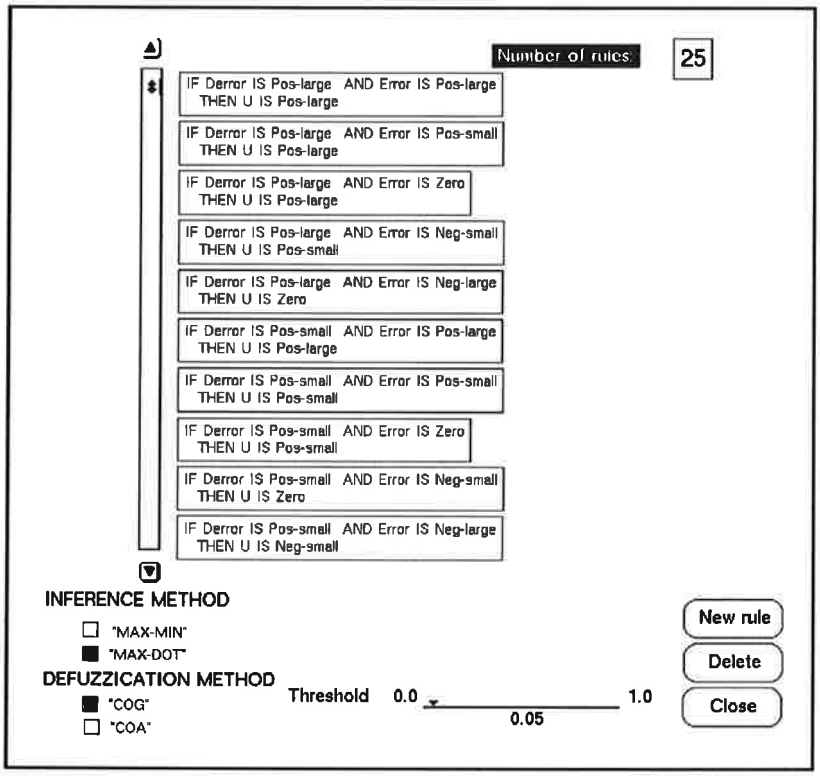
Tillhörighetsfunktionens namn: Vid siffra 5 i figur 3.15. Här står namnet på tillhörighetsfunktionen. Namnet på tillhörighetsfunktionen kommer alltid att vara skrivet rakt över den punkt i tillhörighetsfunktionen med "störst värde i y-led", om flera punkter har samma värde kommer namnet att stå rakt över den punkt som ligger längst till vänster. När man trycker på namnet kommer tillhörighetsfunktionen att överföras till editeringsfönstret.

Fönster nummer 3: Editeringsfönstret för tillhörighetsfunktionerna. När en ny tillhörighetsfunktion flyttas till detta fönster kommer eventuell tillhörighetsfunktion som redan finns i fönstret att "sparas" i sin editerade form.
Det är i detta fönster som man flyttar sina punkter, de svarta fyrkanterna vid 6 och 7 i figur 3.15, så att tillhörighetfunktionerna får önskat utseende. Punkterna kan inte placeras ovanför y-värdet 1 eller nedanför y-värdet 0, ej heller kan en punkt placeras till vänster om x-värdet -1 eller till höger om x-värdet 1, dvs. alla försök att placera punkter i icke definierade positioner kommer att vara omöjligt (i största möjliga mån). Det kommer inte heller att vara möjligt att placera punkterna relativt varandra så att arean under en tillhörighetsfunktions kurva inte är sammanhängande, detta innebär att punkt 7 inte kan placeras till höger om punkt 6 i figur 3.15 ej heller kan punkt 6 placeras till vänster om punkt 7. För att ytterligare minska fel vid körning kan inte den första eller sista punkten till en tillhörighetsfunktion flyttas ifrån y-värdet noll.

3.4 Regeleditorn

Figur 3.16 visar ett öppnat regelblock. Fönstret visar de fuzzyregler som användaren har matat in och sifferfönstret i högra hörnet **number of rules** visar antalet. Med de små piltangenterna kan man bläddra ibland de inmatade reglerna.

Kommandon:



Figur 3.16: Ett öppnat regelblock, regeleditor

Stänga regeleditorn gör man med knappen **CLOSE**. En öppen regeleditor måste stängas innan en ny kan öppnas.

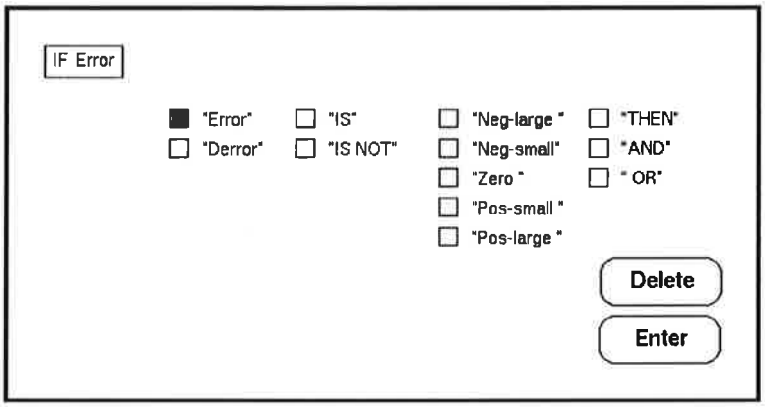
Tröskelvärdet för regulatorn ändras genom att ställa in dragpotentiometern på önskat värde, dvs. till vilken grad en fuzzydelmängd skall vara uppfylld för att den skall tas med vid beräkningen av utsignalen. Tröskelvärdet kan ändras under körning men ändringen genomförs inte förrän regeleditorn stängs.

Inferensmetod bestäms genom att trycka in någon av MAX-MIN eller MAX-DOT under rubriken "Inference method". Man kan ändra på inferensmetod under körning. Körs regulatorn med tabellslagning så måste en ny tabell skapas om man vill byta inferensmetod.

Defuzzieringsmetod väljs genom att trycka in någon av COG eller COA under rubriken "Defuzzication method". Man kan ändra på defuzzifieringsmetod under körning. Körs regulatorn med tabellslagning så måste en ny tabell skapas om man vill byta defuzzifieringsmetod.

Ta bort en regel genom att markera, klicka på, den regeln som skall tas bort. Markerad regel visas med inverterad text. Tryck på **DELETE**.

Programmet ber dig bekräfta borttagningen. Regler kan inte tas bort under körning.



Figur 3.17: Inmatning av en ny regel

En ny regel kan endast skapas då regulatorn är frånslagen. Tryck på **NEW RULE** och ett nytt fönster kommer upp, se figur 3.17. En regel byggs upp av villkor och styråtgärder. Villkor binds samman med **OR** eller **AND**.
Uttryck med flera villkor evalueras från vänster till höger.
Villkoren avslutas med **THEN** därefter följer styråtgärden. Flera styråtgärder kan kopplas sammman med **AND**. När man har valt fuzzyvariabel så visas fuzyvariabelns fuzzymängder. Med **DELETE** suddas den senaste knapptryckningen. Med **ENTER** avslutas inmatningen. Det går inte avbryta inmatningen om endast en del av en komplett regel har matats in.

3.4.1 Regulatorn körs

Genom att välja menyvalet **colour rule on** för den statiska fuzzydelen kommer reglerna i regeleditorn att få olika färger beroende på reglernas uppfyllnadsgrad. Färgernas betydelse :

Färg	**Uppfyllnadsgrad**
Röd	> 0.8
Orange	> 0.5
Gul	> 0.2
Vit	> 0.0
Grå	$\equiv 0.0$

Färgningen stjäl processorkraft och kan därför stängas av med menyvalet **colour rule off**.

3.5 Hur en regulator byggs

Efter det att man skapat ett fönster där man vill placera sin regulator öppnar man fönstret som heter **TOOL-BOX-WORKSPACE**.
På **TOOL-BOX-WORKSPACE** finns alla byggblock som man får använda för att bygga upp sin regulator. Där finns fuzzy-regulatorer[1] och statiska-fuzzy-regulatorer[2] med olika antal ingångar och utgångar plus att det finns adderare, subtraherare, summerare, loop-block, fördröjnings-block m.fl. Det finns även fyra knappar på fönstret.

1. Välj en fuzzyregulator med önskat antal in- och utgångar. Genom att klicka på ett objekt skapas en kopia av objektet. Den kopierade fuzzy-regulatorn flyttas till ett nyöppnat fönster. Därefter öppnar man fuzzy-regulatorn.
   De block som redan finns i den öppnade regulatorn får *inte* flyttas ifrån fönstret, tas bort eller på annat sätt avlägsnas ifrån fönstret (dock får de flyttas inom fönstret).

2. I den öppnade regulatorn lägger man nu in den statiska-fuzzy-regulatorn, som man duplicerat genom att klicka på den, med önskat antal in och utgångar.

3. Nätet som omger den statiska-fuzzy-regulatorn byggs upp med hjälp av de block som finns på **TOOL-BOX-WORKSPACE**. Även dessa block dupliceras när man klickar på dem, efter det att de är duplicerade flyttas de till den öppnade fuzzy-regulatorn.

4. När alla blocken är utplacerade ansluts in- och utgångarna på blocken till önskat nät. Observera att eventuella återkopplingar i nätet måste göras med ett loopblock.

5. Därefter öppnar man den statiska-fuzzy-regulatorn. *Inga* ändringar får göras i detta fönster.

6. Nu skall fuzzy-variablerna och reglerna initieras. Först initieras fuzzy-variablerna genom att man öppnar dem och väljer lämplig konfiguration, enligt sektion 3.3. Efter det att alla fuzzy-variablerna är initierade öppnar man regeleditiorn. Regeleditorn beskrivs i sektion 3.4.

7. När alla fuzzyvariabler har initierats och reglerna är inskrivna kan fuzzyregulatorn startas med menyvalen **control on** eller **control on table**.

[1] Blocket på **TOOL-BOX-WORKSPACE** i vilket det står **OFF**.

[2] Blocket på **TOOL-BOX-WORKSPACE** i vilket det står **FUZZY** och som dessutom har en ritad kurva

De fyra knapparna på fönstret är:

Close: Stänger fönstret.

Stop simulation: Stänger av simuleringen.

Start simulation: Sätter igång simuleringen.

Restart exec-fuzzy: Återstartar regulatorn.
Om regulatorn av en eller annan anledning skulle hänga sig, t.ex. p.g.a. att man går in i och ändrar något objekt samtidigt som regulatorn kör kan man försöka att göra en omstart av regulatorn genom att trycka på denna knapp.

3.6 Hur regulatorn sparas

Man sparar sin regulatorkonstruktion på samma sätt som man sparar G2-program i vanliga fall, dvs. man klickar på menyvalet **Save KB**.
När man sparar designen sparas *allt*, både de ursprungliga definitionerna för fuzzyregulatorn och de nya definitionenar som användaren har gjort. Observera att den tabell som måste skapas vid körning av regulatorn med tabellslagning inte sparas när man sparar designen, utan måste skapas igen.

Även om man använder **Reset** eller **Restart** i G2 så finns regulatordesignen kvar.

Att ladda in sin sparade konstruktion görs på vanligt sätt genom att trycka på menyvalet **Load KB** och ange namnet på den sparade konstruktionen. När man laddar in programmet kommer den eventuellt sparade konstruktionen att bli tillgänglig med det utseende som den hade vid det sparade tillfället.

Kapitel 4

Implementering

Detta kapitel beskriver hur implementeringen av fuzzyregulatorn är gjord och hur fuzzymängder och fuzzyregler representeras. Det förutsätts att läsaren har grundläggande kunskaper om G2 [10]. De namn som används i G2 skrivs med avvikande stil t.ex. **Variabelnamn**.

4.1 Inledning

G2 Version 3.0 tillåter att objekt skapas och manipuleras dynamiskt, även attributen i objekten kan ändras dynamiskt och detta utnyttjas genomgående i vår implementering. Andra nyheter i Version 3.0 som utnyttjas är array hantering och charts.

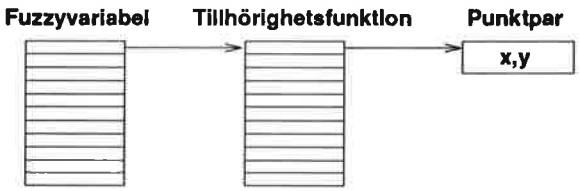
Ett av kraven på examensarbetet var att en fuzzyregulator skulle vara generisk, d.v.s. det skulle vara lätt att skapa nya instanser av regulatorn. Med detta krav försvinner möjligheten att relatera de olika objekten till varandra genom att namnge dem. Istället är blocken i regulatorn ihopkopplade med relationer, via fysisk placering på skärmen, listor och matriser.

Det är mycket svårt att dokumentera en G2-applikation. Uppdelningen på objektdefinitioner, procedurer, relationer, regler m.m. gör en implementering svåröverskådlig. I detta kapitel redovisas bara de viktigaste delarna. Alla definitioner finns på fönstret **Sourcecode-fuzzy-toolbox**.

4.2 Hur fuzzyvariablerna representeras

En fuzzyvariabel är en lista av tillhörighetsfunktioner. Tillhörighetsfunktionerna består av en lista med punktpar, Se figur 4.1. Punktparen definierar

funktionen i x och y led. Klassen för en fuzzyvariabel är **A-fuzzy-variable**.



Figur 4.1: Datastrukturen för en fuzzyvariabel

Här följer en presentation av de ingående klasserna.

A-fuzzy-variable

Supererior class	**item-list**
Attributes	**A-fuzzy-variable-name is ""** **min-position-in-mode is -1** **max-position-in-mode is 1** **number-of-samples is 20** **scalefactor is 1.0** **text-representation is ""**
Default settings	**Element type for item-list : u-ship**

A-fuzzy-variable är en lista av tillhörighetsfunktioner som är representerade av instanser av klassen **u-ship**.
A-fuzzy-variable-name innehåller variabelns namn.
Min-position-in-mode och **max-position-in-mode** bestämmer variabelns normeringsområde.
Number-of-samples anger hur många kvantiseringsnivåer som tillhörighetsfunktionen skall representeras i.
Scalefactor är skalfaktorn som används vid normering.
Text-representation är en textkonstant där alla punktpar och tillhörighetsfunktionsnamn sparas. Detta medför att värdena finns kvar även om **Reset** eller **Restart** används.

u-ship

Superior class	**object**
Attributes	**u-ship-function is an instance of a** point-pair-list **u-ship-name is given by a text-parameter** **sample-function is an instance of a float-array** **degree-of-fulfillment is given by a float-parameter**

U-ship-function är en punktparlista. Dessa punktpar definierar funktionens utseende.
U-ship-name innehåller tillhörighetsfunktionens namn.
Sample-function är den kvantiserade versionen av tillhörighetsfunktionen.
Degree-of-fulfillment innehåller uppfyllnadsgraden för tillhörighetsfunktionen. Detta attribut används när regulatorn körs.

point-pair-list

Supererior class	**item-list**
Attributes	
Default settings	**Element type for item-list : point-pair**

Class name	**point-pair**
Superior class	**object**
Attributes	**x is given by a float-parameter** **y is given by a float-parameter**

Klassen **A-fuzzy-variable** används för att definiera två nya klasser, **Fuzzy-variable-input** och **Fuzzy-variable-output**. Det är instanser av dessa klasser som finns på fönstret till den öppnade statiska fuzzydelen.

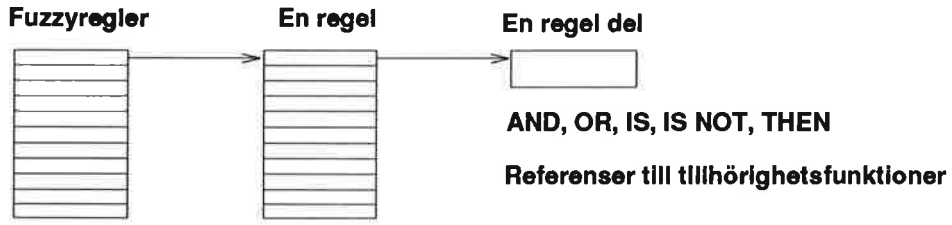
Class name	**fuzzy-variable-input**
Superior class	**a-fuzzy-variable**
Attributes	

Class name	**fuzzy-variable-output**
Superior class	**a-fuzzy-variable**
Attributes	**fuzzy-values is an instance of a float-array** **output-value is given by a float-parameter**

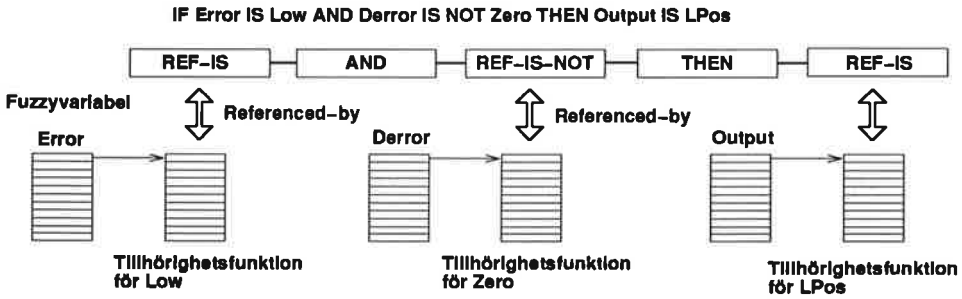
fuzzy-values är en vektor som innehåller utsignalens fuzzyvärde. **output-value** innehåller utsignalens exakta värde. Detta värde har beräknats från **fuzzy-values** med en av defuzzifieringsmetoderna.

4.3 Hur reglerna representeras

Fuzzyreglerna är en lista bestående av enskilda regler. En regel består av en lista av de regeldelar som bygger upp en regel. Se figur 4.2. Referenserna till fuzzyvariablerna etableras genom en relation **referenced-by** som knyts mellan regeldelen **ref-is** eller **ref-is-not** och en tillhörighetsfunktion, **u-ship**, se figur 4.3.



Figur 4.2: Datastrukturen för fuzzyreglerna



Figur 4.3: Datastrukturen för en regel

Class name	**fuzzy-rules**
Supererior class	**item-list**
Attributes	
Default settings	**Element type for item-list : one-rule-list**

Class name	**one-rule-list**
Supererior class	**item-list**
Attributes	
Default settings	**Element type for item-list : rule-element**

Class name	**rule-element**
Supererior class	**item-list**
Attributes	**rulepart has values g2,ref-is,ref-is-not, f-or, f-then, or f-and and is given by a symbolic-parameter**

Class name	**Rule-Editor**
Supererior class	**Message-handler**
Attributes	**fuzzyrules is an instance of fuzzy-rules Text-representation is ""**

Fuzzyrules innehåller fuzzyreglerna, se beskrivningen av klassen **fuzzy-rules**.
Text-representation är en textkonstant som innehåller en kopia av alla reglerna. Detta medför att reglerna finns kvar även om **Reset** eller **Restart** används.

För presentationen av de inmatade reglerna har vi utnyttjat ett existerande meddelandesystem. Meddelandesystemet ligger under **utilities** på **SourceCode-fuzzy-toolbox**-fönstret. Systemet är egentligen gjort för att hantera larmmeddelande och vissa smärre ändringar har genomförts för att anpasssa systemet till vår applikation. Klassen **messagehandler** ifrån meddelandesystemet används för att skapa klassen **Rule-Editor** i fuzzyregulatorn. Det är en instans av denna klass som är synlig på fönstret till den öppnade statiska fuzzydelen. **Rule-Editor** finns i ett antal olika upplagor med olika antal in- och utgångar.

4.4 Lagring av tillhörighetsfunktioner och regler

I programmet är en regulator uppbyggd av relationer mellan tillhörighetsfunktionerna och fuzzy-reglerna, mellan fuzzy-variabler och tillhörighets-

funktioner, listor bestående av tillhörighetsfunktioner, listor med regel-element o.s.v. När man stannar G2 försvinner alla relationer och alla listor töms. För att kunna spara undan regulatorn måste man därför representera all viktig data i konstanter, eftersom konstanter behåller sina värden när G2 stoppas.
I programmet sparas den väsentligaste informationen hos regler och tillhörighetfunktioner i konstanten **text representation** som är en textsträng. Varje **rule-editor** och **fuzzy-variable** har sin egen **text representation**. Här beskrivs endast hur en tillhörighetfunktion läggs in i **text representation** (reglerna sparas på liknande sätt).

Varje gång man stänger ner inmatningsfönstret för en fuzzy-variabel kommer namnen och koordinaterna för tillhörighetsfunktionerna att läggas in i textsträngen till fuzzyvariabeln. All information är sparad i en lång sträng med de olika delarna skilda ifrån varandra med ett antal olika tecken. Mellan namnet och de första koordinaterna finns tecknet (:) och mellan varje koordinatpar finns det ett (,). Varje tillhörighetsfunktion avslutas med ett (!), se nedan.

För en fuzzy-variabel med tre tillhörighetsfunktioner ser **text representation** ut som nedan.

"Neg:-1.0,0.0;-1.0,0.0;-1.0,-1.0;0.5,1.0;0.0,0.0;!Zero:-1.0,0.0;...;!"

När man sparar G2 med **Save KB** kommer informationen om tillhörighetsfunktionerna att ligga som en textsträng i respektive fuzzy-variabel.

När man trycker på **Start**, efter det att man laddat in programmet, kommer textsträngen att packas upp. Vid uppackningen kommer listor och relationer för fuzzy-variabeln att sättas upp igen.

Med hjälp av skiljetecknen (:), (,) och (!) vet uppackningsproceduren vad som skall finnas inom varje del av textsträngen.
Uppackningsproceduren klarar inte av blanktecken i textsträngen. Detta är anledningen till att man inte får ha blanktecken i namnet på tillhörighetsfunktionerna.

4.5 Den kompletta fuzzyregulatorn

Klassen för en fuzzyregulator heter **Fuzzy-controller-x-y**, där x och y står för antalet ingångar respektive utgångar.

Class name	**Fuzzy-controller**
Supererior class	**object**
Attributes	**execution-order is an instance of fuzzy-object-list** **sampling-time is 2** **counter is 1** **modes has values on, off (default is off)**

Execution-order är en lista som innehåller de olika blocken i regulatorns inre struktur : adderare, fördröjningsblock etc. Hur denna lista byggs upp beskrivs nedan.
Sampling-time är en konstant som bestämmer regulatorns samplingstid och kan ändras av användaren.
Counter är en räknare som initieras med värdet ifrån **sampling-time** och som sedan räknas ner en gång i sekunden. Räknaren används av programmet för att avgöra när reglering skall ske. Om räknaren av någon anledning slutar att räkna ner tyder detta på att exekveringsrutinen för fuzzyregulatorerna har stannat. Med knappen **Restart exec-fuzzy** på fönstret **toolbox-workspace** kan man försöka att starta upp exekveringsrutinen igen.
Modes bestämmer om regleringen är av eller på och detta bestäms via menyval.

4.5.1 Att exekvera block i rätt ordning

En fuzzyregulator har ett antal in- och utgångar. Inuti fuzzyregulatorn bestämmer användaren själv utseendet på regulatorn med ett antal block som kan kopplas samman på olika sätt. Den översta klassen **fuzzy-controller** innehåller en exekveringslista. Denna lista byggs upp när regulatorn startar och bestämmer i vilken ordning exekveringen av de olika blocken skall ske.

För regulatorn i figur 3.2 ser exekveringslistan ut enligt följande:

an inport
an inport
a sub
a delay
a sub
a static-fuzzy-controller-2-1
an outport

Proceduren **Determine-execution-order** skapar exekveringslistan. Först i listan läggs alla inportar och loop-block. Utsignalen från dessa block är given utan någon beräkning. Därefter läggs blocken in i ordning så att ingångarna på det block som läggs in sist i listan är anslutna till block som redan finns i listan. Allra sist i listan kommer naturligtvis utportarna. På detta sätt kommer en signal på inporten att föras från block till block tills den når utporten och varje block kommer att utföra sin behandling av signalen.

4.6 Hur ett aritmetiskt block är uppbyggt

Med aritmetiska block avses de block som bygger upp regulatorns inre struktur, dvs. adderare, fördröjning etc. Den abstrakta datatypen för dessa block är gjord för maximalt fyra ingångar och fyra utgångar. Klassen **Fuzzy-object** utgör grunden för aritmetiska block. Klassen **Arithmetric-object-X-Inputs-X-Output** används när man bygger upp en ny klass av aritmetiska block.

Ledningarna som knyter samman de olika blocken i regulatorn är av klassen **fuzzy-connection** som helt överrenstämmer med den vanliga klassen **connection** utom i det att **fuzzy-connection** har ett attribut **val**. **Val** är värdet på signalen i ledningen mellan blocken.

Datastrukturen är lika för alla block. För att beskriva ett aritmetiskt blocks strukturella uppbyggnad följer här en ingående beskrivning av hur additionsblocket är uppbyggt.

Additionsblocket har ett utseende enligt figur 3.3 med två ingångar och en utgång. Blockets uppgift är att addera signalerna på de båda ingångarna med varandra och lägga ut summan på utgången.

$$U_{ut}(t) = U_{in1}(t) + U_{in2}(t)$$

Vid beskrivningen av additionsblocket kommer vi först att beskriva den översta huvudklassen och fortsätta neråt i klasshierarkin till ett additionsblock.

Den översta klassen för alla block i vårt program är **Fuzzy-object**.

Class name	**Fuzzy-object**
Supererior class	**object**
Attributes	**input-proc is input-default** **exec-proc is exec-default** **output-proc is output-default** **in1 is given by a quantitative-parameter** **in2 is given by a quantitative-parameter** **in3 is given by a quantitative-parameter** **in4 is given by a quantitative-parameter** **out1 is given by a quantitative-parameter** **out2 is given by a quantitative-parameter** **out3 is given by a quantitative-parameter** **out4 is given by a quantitative-parameter**

Class name och **Superior class** behöver ingen närmare beskrivning, däremot är förståelsen för de olika procedurerna och parametrarna under **Attributes** viktig för vidare beskrivning av add-blocket.

När en regulator körs kommer **Fuzzy-object** och alla dess underklasser att köra de tre procedurerna vars namn finns i **input-proc**, **exec-proc** och **output-proc**.
Varje underklass till **Fuzzy-object** kommer att ha sina egna procedurer som exekveras vid körning t.ex. har add-blocket **input-proc is input-1-in, exec-proc is exec-add** och **output-proc is output-1-out**

input-proc: När man kör **input-proc** för respektive klass kommer värdet på ingångx på det specifika blocket att överföras till parametern inx där x står för numret på ingången.

exec-proc: Detta är denna procedur som utför blockets behandling av insignalen/insignalerna. I additionsblocket adderar proceduren värdet av parametrarna in1 och in2 och resultatet tilldelas parametern out1.

output-proc: Denna procedur överför värdet på parametrarna outx till utgångx för alla blocket utgångar. X står för numret på utgången.

Anledningen till att alla fyra in- och utgångarna är definierade, trots att

inte alla används i alla underklasser, är att annars uppstår det felmedelande vid inladdning (i form av **inconsistence** i vissa procedurer).

Den första underklassen till **Fuzzy-object** i additionsblockets "träd-struktur" är objektet **arithmetric-object-2-inputs-1-output**.

Class name	**arithmetric-object-2-inputs-1-output**
Supererior class	**fuzzy-object**
Attributes	**input-proc is input-2-in** **output-proc is output-1-out**

Detta objekt har ingen **exec-proc** eftersom det inte finns något block som representerar detta objekt. Däremot har objektet en **input-proc** och en **output-proc.**

Utseendet för **input-proc**-proceduren är.

input-2-in(f: class fuzzy-object)
begin
conclude that the in1 of f = the val of the fuzzy-connection connected at inport1 of f;
conclude that the in2 of f = the val of the fuzzy-connection connected at inport2 of f;
end

Proceduren tilldelar värdet på inport1 till parametern in1 och värdet på inport2 till parametern in2.

På liknande sätt fungerar **output-proc.**

output-1-out(f: class fuzzy-object)
begin
conclude that the val of fuzzy-connection connected at the outport1 of f = the out1 of f;
end

Output-proc tilldelar värdet av out1 till **val** på **fuzzy-connection** som är kopplad till **outport1.**

Nästa objekt i ordningen är **add-class.**

Class name	**add-class**
Supererior class	**artithmetric-object-2-inputs-1-output**
Attributes	**exec-proc is exec-add**

Add-class ärver de två procedurerna **input-2-in** och **output-1-out** ifrån **artithmetric-object-2-inputs-1-output**. Det enda nya attributet är exekverings-proceduren till **add-class**:en.

Utseendet på **exec-proc** till **add-class**.

exec-add(f: class fuzzy-object)
begin
conclude that the out1 of f = the in1 of f + the in2 of f;
end

Proceduren adderar värdena från in1 och in2 med varandra och tilldelar out1 resultatet av additionen.

4.7 Den statiska fuzzydelen

Den statiska fuzzydelen är ett av de aritmetiska blocken. Klassen för den statiska fuzzydelen heter **Static-fuzzy-controller** och finns i ett antal olika upplagor med olika antal in- och utgångar.

Class name	**Static-fuzzy-controller**
Supererior class	**Fuzzy-object**
Attributes	**input-proc input-x-in** **output-proc output-x-out** **exec-proc exec-static-fuzzy** **inference-method has values max-min or max-dot** **threshold is given by a float-parameter** **defuzzification-method** **runtime-table is an instance of a fuzzy-table** **using-table is given by a l-parm-false** **controller-modified is given by a l-parm-true**
inherited atttributes	**in1 is given by a quantitative-parameter** **out1 given by a quantitative-parameter**

Input-proc, output-proc innehåller namnen på procedurerna som sköter hanteringen av signalerna till och från blocket.
Exec-proc innehåller namnet på den procedur som utför fuzzy-algoritmen. Denna procedur beskrivs nedan.

Inference-method bestämmer vilken inferensmetod som skall användas.
Threshold bestämmer till vilken grad en tillhörighhetsfunktion måste vara uppfylld för att den skall tas med vid evalueringen av reglerna.
Defuzzification-method bestämmer vilken defuzzifieringsmetod som används.
Runtime-table innehåller den uträknade tabellen då regulatorn körs med tabellslagning.
Using-table är en logisk variabel som är sann då tabellslagning används.
Controller-modified är en logisk variabel som är sann om något i den statiska fuzzydelen har ändrats sen tabellen skapades. Variabeln används för att avgöra om en ny tabell skall skapas.
in1, ... och **out1**, ... är in- och utsignalerna till den statiska fuzzydelen.

4.7.1 Exekvering av den statiska fuzzydelen

Proceduren som utför fuzzy-algoritmen heter **exec-static-fuzzy**. Om regulatorn körs med tabellslagning så anropas proceduren **look-up-table** under reglering. När regulatorn startas för reglering med tabellslagning så anropas proceduren **create-table**, som skapar tabellen, om det är första gången som tabellslagning används eller om något i regulatorn har ändrats sedan tabellen senast skapades.

Om reglering skall ske direkt så kommer **exec-static-fuzzy** att anropa följande procedurer :

Input-fuzzy anropar proceduren **fuzzication** med fuzzyvariablerna som finns på regulatorns ingångar. Proceduren **fuzzication** normerar insignalen och beräknar därefter uppfyllnadsgraden för tillhörighetsfunktionerna i fuzzyvariablerna.

Reset-dof nollställer uppfyllnadsgraden för de olika tillhörighetsfunktionerna i fuzzyvariablerna på utgången.

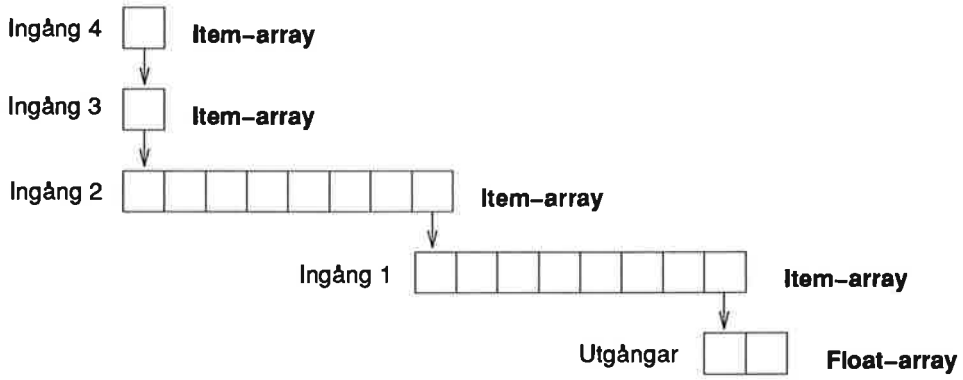
Evaluate-rules går igenom alla reglerna och beräknar uppfyllnadsgraden för tillhörighetsfunktionerna i fuzzyvariablerna på utgången.

Output-fuzzy anropar procedurerna **inference** och **defuzzication**. Proceduren **inference** utför inferens enligt max-min eller max-dot metoden. Proceduren **defuzzication** utför defuzzifiering enligt någon av metoderna COG eller COA.

4.7.2 Hur regulatortabellen är uppbyggd

För att förenkla proceduren, **look-up-table**, som läser i tabellen då regulatorn körs, är tabellen alltid uppbyggd för fyra ingångar. Se figur 4.4. Antalet element i vektorerna för ingångarna är lika med antalet kvantiseringsnivåer som bestämts i fuzzyvariablerna.
Om en fuzzyvariabel har n kvantiseringsnivåer och normeras med gränserna min och max skapas insignalen för elementen i vektorn enligt:

$$insignal = min + \Delta i, \quad i = 0...n-1 \text{ där } \Delta = \frac{max-min}{n}$$



Figur 4.4: Tabell för en statisk fuzzydel med 2 ingångar och 2 utgångar

4.8 Hur nya block skapas

När man skapar nya block i programmet måste de nya blocken uppfylla vissa krav. Vilka krav som gäller anges nedan under respektive huvudgrupp av block.

4.8.1 Att skapa nya aritmetiska block

Med de aritmetiska blocken menas additions-block, loop-block, fördröjningsblock m.fl.

1. Skapa ett nytt objekt och namnge objektet.
2. Sätt **Superior class** till **arithmetric-object-x-inputs-y-output** där x och y står för antalet ingångar respektive antalet utgångar på det aritmetiska-blocket. Om det bland de redan definierade klasserna **arithmetric-object-x-inputs-y-output** inte finns någon klass med

rätt konfiguration måste man själv skapa ett nytt objekt med rätt utseende se 2a. Om det däremot finns ett block med rätt utseende fortsätt med punkt 3.

(a) För att skapa ett nytt **arithmetric-object-x-inputs-y-output** skapar man ett nytt objekt och namnger det nya blocket. För enkelhetens skull, namnge gärna objektet med **arithmetric--object-x-inputs-y-output** och fyll i rätt antal in- och utgångar i namnet istället för x och y.

(b) Sätt **Superior class** på det nya blocket till **Fuzzy-object** och i **Attributes specific to class** anges **input-proc is input-x-in** där x står för antalet ingångar på det aritmetiska blocket. Ange även **output-proc is output-x-out** (x är lika med antalet). Upp till 4 in- och utgångar finns implementerade (om man vill ha fler använd **Inspect, go to input-4-in/output-4-out** och skapa en procedur som är uppbyggd på samma sätt).

3. Efter det att man har skrivit in **Superior class** anger man i **Attributes specific to class** vilken procedur som skall utgöra exekveringsproceduren, kommando strukturen är **exec-proc is zzz** där *zzz* är den procedur som innehåller vad som skall göras med de olika signalerna i blocket man skapat.

4. Nästa steg är att lägga in **Stubs** för önskat antal in- och utgångar. Kommando-strukturen för detta är **an input fuzzy-connection inportx located at nn** eller för en utgång
an output fuzzy-connection outportx located at nn där x i båda fallen står för numret på in- eller utgången och där nn står för den fysiska placeringen av porten på blocket.

5. Gör **create instance** på objekt-definitionen för att få det nya blocket och placera instansen på **Tool-Box-Workspace**.

4.8.2 Att skapa nya Fuzzy-controller-block

De block som redan är implementerade är (ingångar-utgångar) 2-1, 3-1, 3-2, 4-1, 4-2 och 4-3. Hur nya konfigurationer skapas beskrivs nedan.

1. Ändra **G2 user mode** till **administrator**.

2. Skapa ett nytt objekt och namnge det.

3. Sätt **Superior class** till **Fuzzy-controller**.

4. I **Attributes specific to class** fyller man i alla in- och utgångsvariabler.
   Strukturen för ingångarna är **inx is an instance of a fast-quantitative-variable** där x står för numret på ingången.
   Strukturen för utgångarna är **outx is an instance of a quantitative-parameter** där x står för numret på utgången.

5. Därefter fyller man i hur man vill ha sina **Stubs**. Strukturen för en inport är **an input connection inportx located at left nn**, x står för ingångsportens nummer och nn den fysiska placeringen. Utgångens struktur är **an output connection outportx located at right nn**, x står för utgångsportens nummer och nn den fysiska placeringen.

6. Nästa steg är att skapa en instans av objektet. Till instansen skapar man ett **subworkspace**.

7. Kopiera inportarna (av klassen **inport-class**) som ligger på **Tool-Box-Workspace** tills antalet inportar stämmer med antalet **Stubs**. Gör därefter samma sak med utportarna (av klassen **outport-class**). Det är rekommendabelt att även lägga in en **hide workspace button** på fönstret.

8. Välj **table** på varje inport och utport. I varje inport och utport skall **Attribute name** skrivas in med strukturen **inx** respektive **outx** med x som anger till vilken **Stubs** på **Fuzzy-controller**:n som in- eller utporten kommer vara ihopkopplad med.

9. Lägg den skapade **Fuzzy-controller** på **Tool-Box-Workspace** och kopiera det nu färdigskapade "orginalet" från detta fönster för att få ett nytt användbart **Fuzzy-controller**-block på något annat fönster.

10. Ändra **G2 user mode** till **user**.

4.8.3 Att skapa nya Static-fuzzy-controller-block

De block som redan är implementerade är (ingångar-utgångar) 1-1, 2-1, 3-1, 3-2, 4-1, 4-2 och 4-3. Hur nya block skapas beskrivs nedan.

1. Ändra **G2 user mode** till **administrator**.

2. Skapa ett nytt objekt och namnge det.

3. Sätt **Superior class** till **Static-fuzzy-controller**.

4. I **Attributes specific to class** fyller man i **input-proc is input-x-in** och **output-proc is output-x-out**, x är antalet in- respektive

utgångar.
X = 1 – 4 är implementerat i input- och output-proc. Om man vill ha fler in- eller utgångar ta **Inspect, go to input-4-in** för ingångarna och **Inspect, go to output-4-out** för utgångarna, skapa sedan en procedur som är uppbyggd på samma sätt. Skriv även in antalet utgångar enligt **nr-of-outputs is x** där x är antalet utgångar.

5. Ange därefter namnens fysika placering i förhållande till blocket, för in- och utgångarna, vid rubriken **Attribute displays**. Strukturen för detta är **inx-name offset by** (n_x, n_y), x står för ingångsnumret, n_x anger den fysiska placeringen i x led och n_y anger den fysiska placeringen i y led (de båda placeringarna är relaterade till mitten på en instans av objektet). För ett utgångsnamn blir strukturen på liknande sätt **outx-name offset by** (n_x, n_y).

6. **Stubs** fylls i med **an input fuzzy-connection inportx located at left nn** (x är ingången nummer och nn dess fysiska placering). För utgången är koden, på samma sätt, **an output fuzzy-connection outportx located at right nn** (x är utgången nummer och nn dess fysiska placering).

7. Skapa en instans av objektet, som du arbetat med, och skapa ett **subworkspace** till instansen.

8. Ta nu och kopiera **fuzzy-variable-input**, som ligger på **Tool-Box-Workspace**, tills antalet ingångar stämmer med antalet ingångs-stubbar på den skapade **Static-fuzzy-controller**:n. Gör därefter samma sak med **fuzzy-variable-output**:s.

9. Fyll i tabellerna för **fuzzy-variable-input** och **fuzzy-variable-output**.
Fuzzy variable name: Detta namn skall överensstämma med namnet på den in- eller utgång på **Static-fuzzy-controller**:n som **fuzzy-variable** är kopplad till t.ex. **input1**.
Attribute name: Anger till vilken in- eller utgång på **Static-fuzzy-controller**:n som **fuzzy-variable** är kopplad till. Strukturen måste vara **inx** respektive **outx**, x står för numret på ingången respektive utgången.

10. Nästa steg är att kopiera **rule-editor** som finns på **Tool-Box-Workspace**.

11. Koppla nu samman **fuzzy-variable** med **rule-editor** genom att dra ut stubbarna från **fuzzy-variable** och fäst dem till **rule-editor**, det är betydelselöst var på **rule-editor** man ansluter kopplingarna.

12. Det är rekommendabelt att även lägga in en **hide this workspace**-knapp.

13. Lägg den färdiga **Static-fuzzy-controller** på **Tool-Box-Workspace** och kopiera den ifrån detta fönstret.

14. Ändra **G2 user mode** till **user**.

Kapitel 5

Simulering

Fuzzyregulatorn prövades på en process och jämfördes med en konventionell PID regulator. För simuleringen skapades ett antal simuleringsblock som kan användas för att realisera olika överföringsfunktioner. Även några prov med reglering med tabell genomfördes.

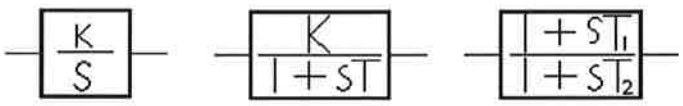
5.1 Processen

Processen tillsammans med designen av fuzzyregulatorn är tagen ifrån [11]. Processen är ett andra ordningens system, icke minfas.

$$G_p = \frac{K(1 - T_1 s)}{(1 + T_1 s)(1 + T_2 s)}$$

$$T_1 = 4s, \quad T_2 = 10s, \quad K = 1$$

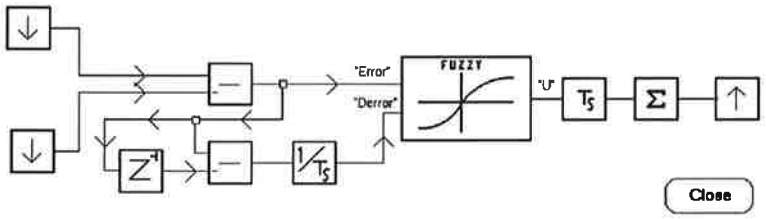
För simulering av processen så används G2:s interna simulator.



Figur 5.1: Simuleringsblock

För simuleringen har vi skapat ett antal block med olika överföringsfunktioner, se figur 5.1. Med dessa block kan olika processer lätt realiseras. Definitionerna till blocken ligger under rubriken **Definitions and objects concerning simulation** på fönstret **sourcecode-fuzzy-toolbox**.

5.2 Fuzzyregulatorn



Figur 5.2: Fuzzyregulatorns struktur

Figur 5.2 visar strukturen på den fuzzyregulator som användes vid simuleringen. Av insignalerna till fuzzyregulatorn skapas reglerfelet e(n) och differensen av reglerfelet de(n) = e(n) - e(n-1). Utsignalen utgörs av summan av fuzzydelens utsignal. Denna struktur motsvarar en PI-regulator. En jämförande studie av fuzzyregulatorer med PI struktur finns i [5] [6].

Konventionell PI regulator på inkrementell form:

$\Delta u_n = K_p(e_n - e_{n-1} + \frac{T}{T_I}e_n)$

utsignalen : $u_n = \Delta u + u_{n-1}$

Fuzzyregulatorn:

$\frac{\Delta u(n)}{K_3} = F\left[\frac{e(n)}{K_1}, \frac{de(n)}{K_2}\right]$

K_1, K_2 och K_3 är skalfaktorerna för fuzzyvariablerna e, de och Δu.

Funktionen F bestäms av fuzzyreglerna, tillhörighetsfunktionernas utseende, inferensmetoden och defuzzifieringsmetoden. Normalt så kommer funktionen F att vara olinjär. Om man jämför den konventionella PI-regulatorn på inkrementell form med fuzzyregulatorn får man fram hur skalfaktorerna, för fuzzyregulatorn, verkar.

Förstärkning: $K_p = \frac{K_3}{K_2}$

Integraltid: $T_I = \frac{K_3}{K_1}$

Betrakta ovanstående formler endast som en vägledning när regulatorn skall trimmas.

Reglerna är uppbyggda enligt :

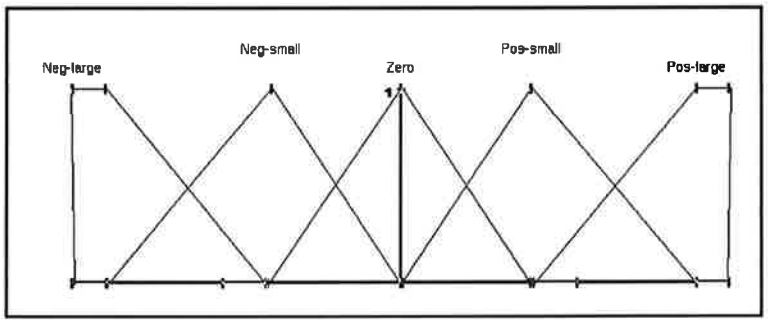
1. Om både e(k) och de(k) är noll så ändra inte utsignalen.
2. Om e(k) kommer att gå mot noll i tillfredsställande takt så ändra inte utsignalen.

3. Om inte e(k) går mot noll så låt styrsignalen bero på storleken och tecknet på e(k) och de(k).

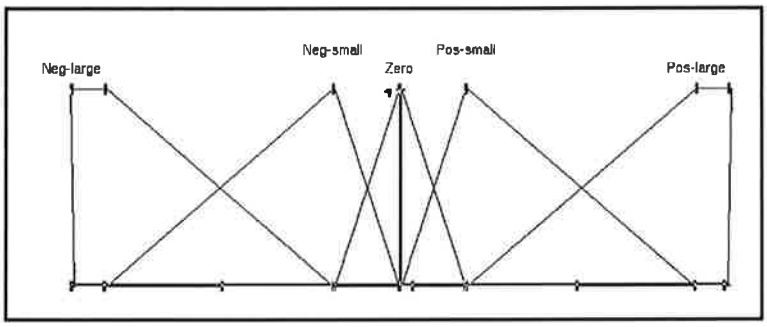
Reglerna som används i fuzzyregulatorn

e(k):/d(k):	NL	NS	Zr	PS	PL
PL	Zr	PS	PL	PL	PL
PS	NS	Zr	PS	PS	PL
Zr	NL	NS	Zr	PS	PL
NS	NL	NS	NS	Zr	PS
NL	NL	NL	NL	NS	Zr

NL=Negative Large, NS=Negative Small, Zr=Zero
PS=Positive Small, PL=Positive Large



Figur 5.3: Tillhörighetsfunktionerna för insignalenerna e(n) och de(n)

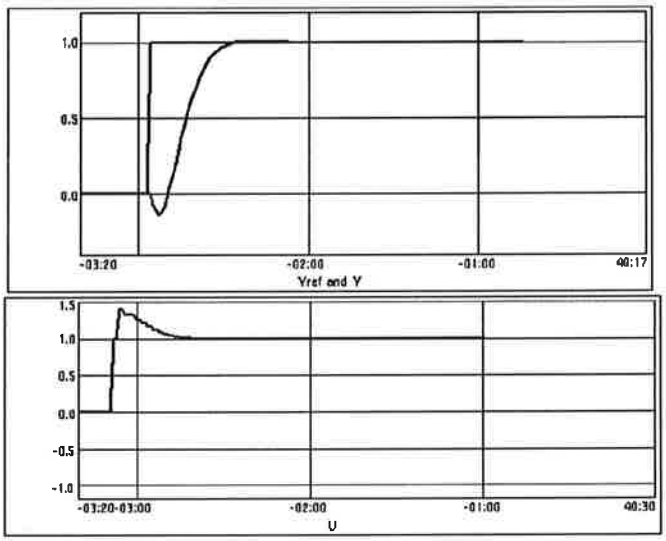


Figur 5.4: Tillhörighetsfunktionerna för utsignalen $\Delta u(n)$

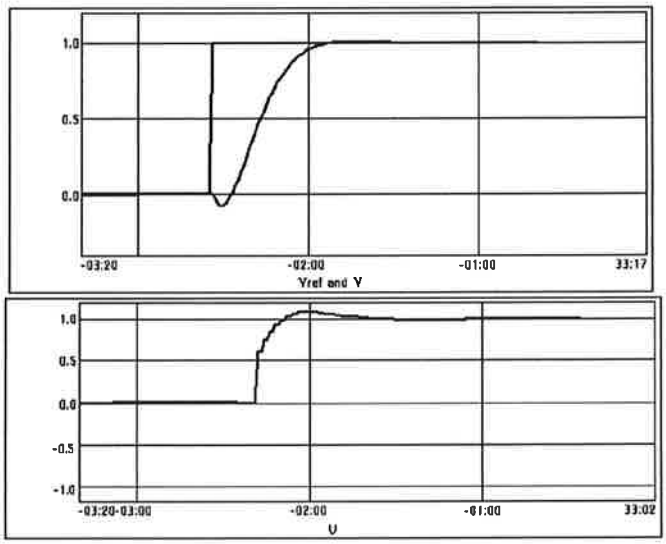
5.3 Simulering

Figurerna 5.5, 5.6 och 5.7 visar stegsvaret för PID, PI och fuzzyregulatorn för processen. Snabbast stegsvar har naturligtvis PID regulatorn. Eftersom fuzzyregulatorn har PI karaktär bör den jämföras med en PI regulator. För

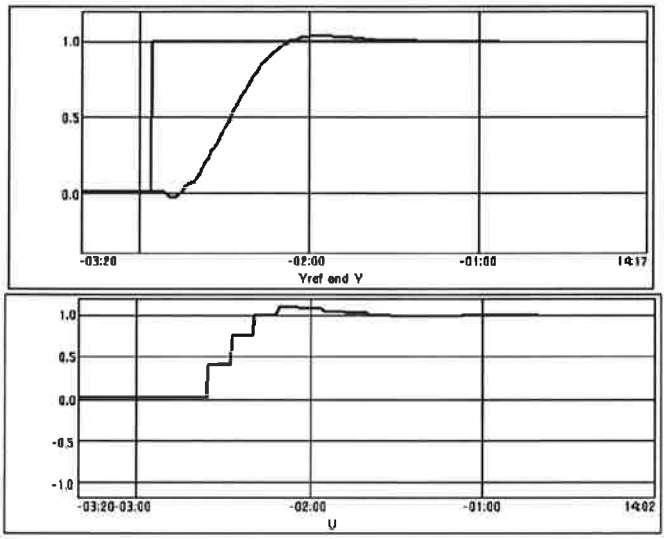
att ställa in PI regulatorn finns många metoder, t.ex. stegsvarsmetoden. För fuzzyregulatorn saknas motsvarande metod. I fuzzyregulatorn finns de tre skalfaktorerna, dessutom kan man ändra på tillhörighetsfunktionernas utseende. Det är möjligt att uppnå samma prestanda som den konventionella PI regulatorn på denna enkla process [5], men det tar längre tid att trimma in fuzzyregulatorn.



Figur 5.5: Stegsvar för PID-regulator



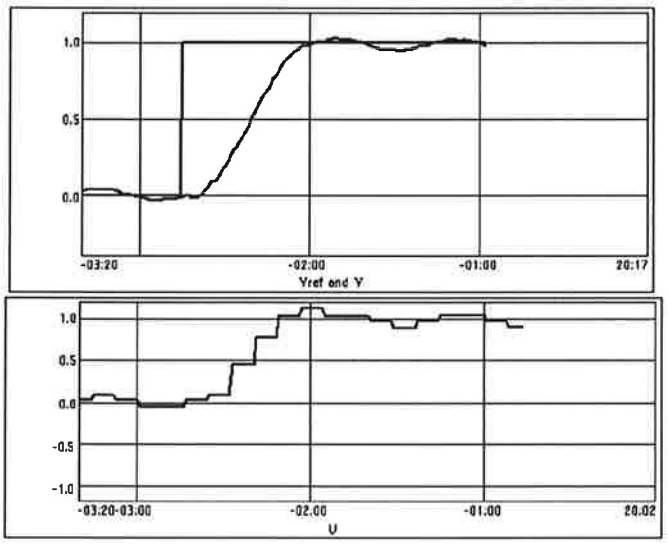
Figur 5.6: Stegsvar för PI-regulator



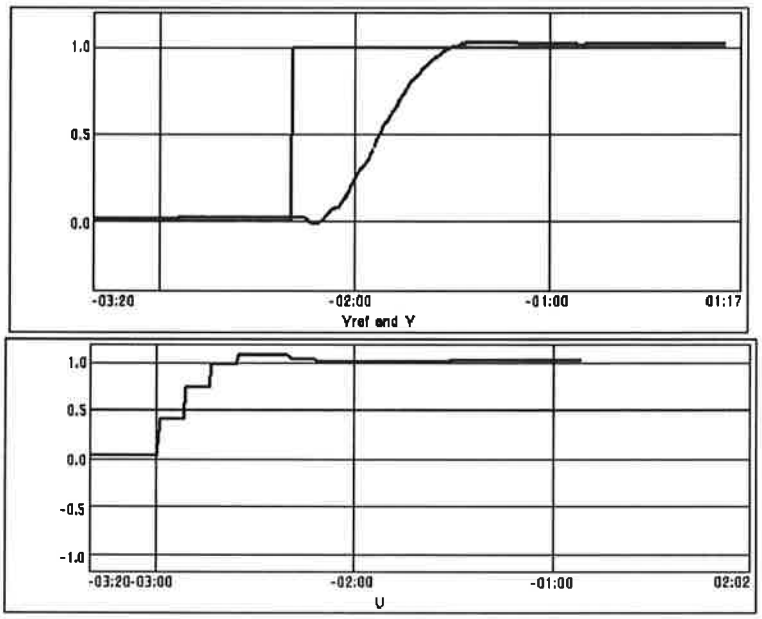
Figur 5.7: Stegsvar för fuzzy-regulator

5.4 Tabellslagning

Figur 5.8 och 5.9 visar stegsvaret för regulatorn när tabellslagning används. Notera ripplet hos signalen i figur 5.8. Ripplet beror på att ingen interpolering sker mellan tabellvärdena. Är tabellen gles medför detta att utsignalen kommer att svänga mellan närliggande värden i tabellen. Genom att förbättra upplösningen i tabellen minskar ripplet, se figur 5.9.



Figur 5.8: Reglering med tabell 15 x 15, notera ripplet



Figur 5.9: Tabell 40 x 40

5.5 Att skapa nya simuleringsblock.

Det finns en punkt som är viktig att tänka på när nya simuleringsblock skapas. För att kunna starta och stoppa simuleringen med knapparna på **TOOL-BOX-WORKSPACE** bör den nya simuleringsprocessen ha **Superior class** satt till **process** eller **signal-generator**. En annan lösning är att i **PROCESS-SIMULATION** under punkten **Items belonging to this model** skriva in namnet eller klassen på den nya processen.

5.5.1 Att tänka på då nya simuleringsblock eller Fuzzy-controller-block skapas.

För att överföra värden mellan blocken vid simulering eller uppkoppling mot omvärlden används **generic-simulation-formula** respektive **generic-formula**.

De generiska-simulerings-formlerna är inskrivna enligt;

```
the in1 of any siso-process s = the first of
  the following expressions that has a value
     (the u of the pid-controller
        connected at the inport of s,
     the out1 of the siso-process
        connected at the inport of s,
     the out1 of the fuzzy-controller
        connected at the inport of s,
     the out2 of the fuzzy-controller
        connected at the inport of s,
     the out3 of the fuzzy-controller
        connected at the inport of s,
     the out4 of the fuzzy-controller
        connected at the inport of s,
     0)
```

De generiska-formlerna är inskrivna enligt;

```
let the in1 of any fuzzy-controller f = the first
  of the following expressions that has a value
     ( the out1 of the signal-generator
        connected at the inport1 of f,
     the out1 of the siso-process
        connected at the inport of f,
     the out1 of the fuzzy-controller
        connected at the inport1 of f,
     the out2 of the fuzzy-controller
        connected at the inport1 of f,
     the out3 of the fuzzy-controller
        connected at the inport1 of f,
     the out4 of the fuzzy-controller
        connected at the inport1 of f,
     0)
```

Som man lätt inser av ovanstående kod måste man skriva om dessa formler, eller skapa nya formler, om t.ex. man har fler än en utgång på ett simulerings-block, man har fler än en ingång på ett simulerings-block, man har fler än fyra utgångar på ett **fuzzy-controller**-block eller man har fler än fyra ingångar på ett **fuzzy-controller**-block.

Kapitel 6

Sammanfattning

6.1 Sammanfattning av examensarbetet

Det två första veckorna av examensarbetet gick åt för att lära oss att programmera i G2. Därefter började vi prova olika lösningar på användargränssnittet. Vi tittade bl.a. på ett kommersiellt CAD paket för fuzzyregulatorer från *Togai*. Viss erfarenhet av att implementera en fuzzyregulator hade vi ifrån ett tidigare projekt i kursen *Datorimplementering av reglersystem*. Att implementera fuzzy-algoritmen är förhållandevis enkelt men fuzzyregulatorn kräver många beräkningar för varje sampel.

Simulering

Vi simulerade en enkel process. Att ställa in en vanlig PI-regulator gick mycket lätt. Fuzzyregulatorn var svårare och tog längre tid att ställa in. Dels har fuzzyregulatorn fler parametrar att ställa in, dels saknas enkla metoder för att få en bra grundinställning. Med hänvisning till ovanstående finns det ingen anledning att byta ut en väl fungerande PID regulator mot en fuzzyregulator.

Vid reglering med tabell får man göra en avvägning mellan minnesåtgång och reglernoggrannhet. Med en gles kvantisering kan det uppstå rippel på processens utsignal. Det bästa antalet kvantiseringar för en specifik fuzzyregulator är mycket intimt sammankopplat med designen av regulatorn, dvs. utseendet på tillhörighetsfunktionerna, skalfaktorernas storlek etc.

6.2 Synpunkter på G2

G2 är en mycket kraftfull utvecklingsmiljö för grafiskt orienterade program. Många funktionerna som krävs för att implementera ett användargränssnitt

finns inbyggda i G2 som t.ex. knappar och diagram. G2 är objektorienterat och tillåter generella konstruktioner av formen: *för alla objekt av klassen nn så gör ...* De olika inbyggda möjligheterna i G2 underlättar programmeringsarbetet avsevärt. Att implementera fuzzyregulatorn i t.ex. C++ hade tagit mycket längre tid. En stor fördel med G2 är att det är lätt att testa en nyskriven procedur, vilket gör att man hela tiden testar små bitar av programmet under implementeringen.

Svaga punkter hos G2

- Procedur-editorn är primitiv. Editorn är anpassad för korta procedurer men G2:s "pratiga kod" leder till att procedurerna lätt blir långa. Editorns dåliga funktion gör att man undviker att kommentera procedurerna.
- I ikon-editorn är det inte möjligt att skriva text, utan varje bokstav måste byggas upp av cirkelbågar, linjer och rektanglar.
- Det är svårt att få en överblick av ett inskrivet program eftersom varje objekt kan påverkas av generella-formler, regler och procedurer. Därför är det också besvärligt att dokumentera en G2-applikation.

6.3 Förslag till vidareutveckling och förbättring

- Rutiner för att skriva ut den skapade regulatortabellen på en fil i lämpligt format, t.ex. i ett format som ett C-program kan läsa. Till detta skall kopplas en program som läser in filen och sedan kan använda tabellen för att implementerara en fuzzyregulator. Program som t.ex. kan skrivas i C skall också kunna genereras automatiskt.
- Rutiner för att interpolera värden vid tabelluppslagning.
- Rutiner för att visa tillhörighetsfunktionernas uppfyllnadsgrad under reglering.
- Förbättra editeringsmiljön för regler och tillhörighetsfunktioner.
  - För tillhörighetsfunktionerna kan man komplettera den grafiskt orienterade editorn med möjligheten att specificera funktionens utseende med hjälp av ett matematiskt uttryck.
  - Regeleditorn kan kompletteras med funktioner för kopiering och editering av befintliga regler.
  - En ny funktion för regeleditorn är automatisk generering av regler. Regeluppsättningen för fuzzyregulatorer med PI, PD och PID karaktär skulle kunna genereras automatiskt.

Litteraturförteckning

[1] L. A. Zadeh: *Fuzzy Sets*, Information and Control, vol 8, sid. 338-353, 1965.

[2] L. P. Holmblad; J. J. Ostergaard: *Control of a Cement Kiln by Fuzzy Logic*, Fuzzy Information and Decision Processes, M. M. Gupta and E. Sanches eds. North Holland, Amsterdam, 1982.

[3] P. J. King och E. H. Mamdani: *The Application of Fuzzy Control Systems to Industrial Processes*, Automatica, vol. 13, sid. 235-241, 1977.

[4] R. M. Tong: *A Control Engineering Review of Fuzzy Systems*, Automatica, vol. 13, sid. 559-569, 1977.

[5] Hao Ying, William Siler och James J. Buckley: *Fuzzy Control Theory: A Nonlinear Case*, Automatica, vol 26, Nr 3, sid. 513-520, 1990.

[6] Kwok L. Tang och Robert J. Mulholland: *Comparing Fuzzy Logic with Classical Controller Designs*, IEEE Transactions on systems, man, and cybernetics, vol 17, nr 6, sid. 1085-1087, 1987.

[7] M. Tanaka, A. Patani,K. Yamaka: *Single Loop Fuzzy Controllers*, Advances in Instrumentation and Control, vol 46, part 2, 1991.

[8] Karl-Erik Årzén: *Computer Implementation of Control Systems*, kap. 12, Institutionen för reglerteknik LTH, 1991.

[9] Chuien Chien Lee : *Fuzzy Logic in Control Systems: Fuzzy Logic Controller* , IEEE Transactions on systems, man, and cybernetics, vol 20, nr 2, sid. 404-434, 1990.

[10] *G2 Reference Manual Version 3.0*, Gensym, 1992.

[11] T. Frutiger: *Fuzzy Logic, Fuzzy regler mit PI ähnlichen Verhalten*, Landis & Gyr Building Control AG, 1992.