CODEN: LUTFD2/(TFRT-5375)/1–54/(1987)

VISIDYN
– Ett program för interaktiv analys av reglersystem

Emil Granbom
Torsten Olsson

Institutionen för Reglerteknik
Lunds Tekniska Högskola
November 1987

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

Document name
MASTER THESIS

Date of issue
November 1987

Document Number
CODEN: LUTFD2/(TFRT-5375)/1-54/(1987)

Author(s)
Emil Granbom
Torsten Olsson

Supervisor
Karl Johan Åström

Sponsoring organisation
STU 87-2503

Title and subtitle
VISIDYN – Ett program för interaktiv analys av reglersystem
(VISIDYN—An interactive program for design of linear dynamic systems)

Abstract

The thesis is made within the Department of Automatic Control in Lund. A program "VISIDYN" is written in C for Sun 3/50. The program is to be used in education, but it can also, with a few extensions, be used as a tool by more experienced designers. VISIDYN is a graphics based system for exploring linear dynamic systems. The basic idea is that a linear dynamic system can be examined from many different points of view. To work with systems it is useful to be able to jump from one viewpoint to another.The key idea is to provide a collection of windows each corresponding to one view of the system and to provide interconnections so that if one view is modified then the other views will automatically follow. The system illustrates the properties of a simple closed loop system. This suggests two views, the open loop and the closed loop. For each of these two views the program shows singularity diagrams, Bode diagrams, and the numeric values of the poles and zeros.

Key words
Bode diagrams; Computer aided design; Interactive computing; Singularity diagram; Transfer function; Windows.

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
54

Recipient's notes

Security classification

Förord

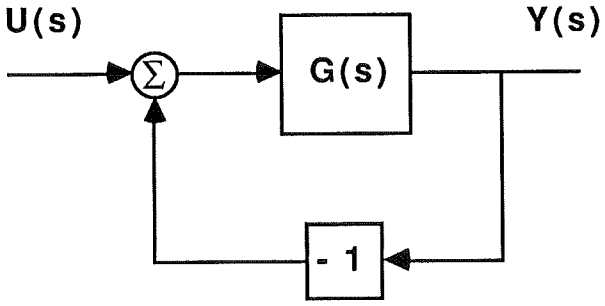
Föreliggande arbete är resultatet av ett examensarbete som har utförts av Torsten Olsson och Emil Granbom, studerande på Datatekniklinjen vid Lunds Tekniska Högskola, LTH, under sommaren och hösten 1987. Idén till arbetet kommer från professor Karl Johan Åström, institutionen för Reglerteknik vid LTH, som även har handlett arbetet. Vi vill även rikta ett tack till Sven-Erik Mattson på Reglerteknik för hans bistånd med vissa matematiska spetsfundigheter, samt till Dan Oscarsson på institutionen för Datalogi vid LTH som har kommit med många tips angående SunView och C.

1 Inledning

VISIDYN är ett grafikbaserat, interaktivt program för att undersöka egenskaperna för lineära dynamiska system. Den grundläggande idén är att ett lineärt dynamiskt system kan ses utifrån många olika vyer. När man arbetar med sådana här system är det till stor hjälp att kunna gå från en vy till en annan på ett snabbt och enkelt sätt. I det här examensarbetet har vi lagt tyngdpunkten på de grafiska aspekterna, och vi har dessutom valt att begränsa oss till system med en insignal och en utsignal, så kallade SISO-system efter engelskans single-input-single-output. VISIDYN är i första hand tänkt som ett undervisningshjälpmedel men programmet kan utvidgas till ett allmänt konstruktionshjälpmedel. I [14] beskrivs ett likartat arbete med delvis annan inriktning.

Tanken har varit att tillhandahålla en uppsättning "fönster" som var och en ger en vy av systemet. Dessa fönster kopplas sedan samman så att en modifiering av systemet i ett fönster leder till en omedelbar modifiering även i de andra fönstren. Här är "omedelbar" ett nyckelord, vi har lagt ner stor möda både vid designen och implementeringen på att få ett system som snabbt reagerar på varje ändring av systemet.

Programmet illustrerar egenskaperna hos ett enkelt återkopplat system enligt figur 1. Detta ger omedelbart förslag till två synsätt, det öppna och det slutna systemet. Vi har valt att koppla samman vyerna parvis så att man alltid har det öppna och det slutna systemet uppe samtidigt. I ett sådant fönsterpar kommer alltid det öppna systemet att ligga till vänster och det slutna systemet till höger.



Figur 1. Enkelt återkopplat SISO-system.

Låt funktionen G(s) vara överföringsfunktionen för det öppna systemet. Den skrivs som

$$G(s) = \frac{b_0 s^m + \ldots + b_m}{s^n + a_1 s^{n-1} + \ldots + a_n} = \frac{B(s)}{A(s)}$$

För att hålla examensarbetet på en rimligt nivå bestämde vi att samtliga koefficienter i A- och B-polynomen ska vara reella. Detta får till följd att samtliga komplexa poler och nollställen kommer att uppträda i komplexkonjugerade par. Dessutom har vi lagt på den begränsningen att

överföringsfunktionerna **inte** får innehålla några tidsförskjutningar, alltså faktorer på formen

$$e^{-sT}$$

Vidare har vi bestämt att överföringsfunktionerna skall vara propra, dvs det ska gälla att grad B ≤ grad A.

Överföringsfunktionen för det återkopplade systemet ges av

$$G_{cl}(s) = \frac{G(s)}{1 + G(s)} = \frac{B(s)}{A(s) + B(s)} \qquad (1)$$

Det öppna systemet erhålles alltså ur det slutna genom

$$G(s) = \frac{G_{cl}(s)}{1 - G_{cl}(s)} = \frac{B(s)}{A(s) - B(s)} \qquad (2)$$

Eftersom både det öppna och det slutna systemet är lineära dynamiska system med en insignal och en utsignal måste vi kunna återge egenskaperna hos sådana system.

Bland de många olika sätt som finns för att åskådliggöra ett reglersystem har vi valt tre:

- singularitetsdiagram
- Bodediagram
- numerisk angivelse av poler och nollställen

Både det öppna och det slutna systemet har karakteriserats på dessa tre sätt. Relationerna mellan singularitetsdiagram, bodediagram och rationella funktioner är uppenbara, för närmare beskrivning av dessa begrepp hänvisas till någon lärobok i reglerteknik, t ex [3] eller [15]. Relationerna mellan öppna och slutna system ges av (1) och (2).

Exempel på utvidgningar är Nyquistdiagram och stegsvar, men den valda uppsättningen är tillräcklig för att illustrera principen. Skulle man vilja lägga till fler vyer i framtiden så är det relativt enkelt. Hur man gör framgår av kapitlet "Att lägga till egna moduler".

2 Planering

Ett antal praktiska frågor måste besvaras innan konstruktionen av systemet kunde påbörjas.

2.1 Maskinvara

Den första, och kanske allra viktigaste frågan som vi hade att ta ställning till var: vilken typ av dator skulle användas? De två kandidater som ansågs realistiska var Macintosh från Apple och arbetsstationer från Sun Microsystems. Det blev inte några större diskussioner om valet. Macintosh uteslöts på grund av att den har så liten skärm.

2.2 Fönsterhanteringssystem

När valet av maskin var klart var nästa steg att bestämma vilket fönstersystem som skulle användas och här fanns det i princip tre alternativ:

- SunView — det fönstersystem som är standard på Sun arbetsstationer
- X Windows — det system som flest tillverkare har enats om att använda
- NeWS — ett nytt system från Sun

Fördelen med SunView är att det är ett fönstersystem som man säkert hittar på alla Sun arbetsstationer, eller för att vara noggrannare på alla arbetsstationer i Sun-3 serien. Samma skäl kan också anges som ett stort minus. Vill man flytta programmet till någon annan dator kan det bli besvärligt.

NeWS (Network extensible Window System) är ett system för grafik och fönsterteknik. Det är utvecklat av Sun Microsystems och är byggt för att kunna arbeta i nätverk. Detta innebär att man klart skiljer mellan fönstersystem och tillämpningsprogram. En annan stor fördel är att det är lätt att få ut sina bilder på laserskrivare eftersom systemet bygger på Postscript. Postscript är det språk som har blivit standard när det gäller att överföra grafik till en laserskrivare. Den stora nackdelen med NeWS är att inga andra tillverkare har visat något större intresse för systemet, bl a därför att Sun håller väldigt hårt på kontrollen av systemet.

X Windows är den största konkurrenten till NeWS och ser ut att bli den nya industristandarden vad gäller fönsterhanteringssystem. Det utvecklades av Massachusetts Institute of Technology (MIT) och stöds nu av ett flertal stora tillverkare av arbetsstationer, bl a Hewlett-Packard och Digital Equipment. Även X Windows är ett system baserat på nätverk men det genererar grafik på ett annorlunda sätt än NeWS. Ytterligare diskussioner om NeWS och X Windows ges i [1] och [6].

Vi valde SunView, trots att det kanske är det sämsta valet. Anledningen till att vi på detta sätt låste oss till Sun, istället för att skriva ett portabelt program, var helt enkelt den att det var det enda som fanns tillgängligt när vi skulle börja. Vi kunde inte vänta tills antingen NeWS eller X Windows hade installerats eftersom ingen visste när detta skulle kunna ske. Nu med facit i

hand vet vi att det bara dröjde ungefär en månad. En annan detalj som löste sig själv i och med valet av SunView var hur användningen av musens tre knappar skulle användas, här har Sun ställt upp rekommendationer som syftar till att få ett enhetligt utseende på de program som använder SunView.

2.3 Programspråk

Vi var också tvungna att bestämma oss för vilket språk vi skulle programmera i. Här valde vi mellan Pascal och C även om SIMULA nämndes i diskussionerna till en början, men SIMULA föll bort av samma orsak som NeWS och X Windows tidigare, det fanns helt enkelt inte tillgängligt på Sun i början av sommaren 1987. Tills slut valde vi att arbeta i C, trots att C har ett rykte om sig att vara ostrukturerat. Anledningarna var flera:

- Det var ett nytt programspråk, för oss båda, som det kan vara bra att kunna
- Sun har UNIX som operativsystem, och C är intimt förknippat med UNIX
- Användningen av SunView krävde i princip att man arbetade i C

3 Allmänt om programmet

Detta kapitel beskriver användningen av programmet i stort. Den övergripande strukturen hos programmet, de moduler som ingår och hur de samverkar beskrivs också.

3.1 Uppstart

VISIDYN måste köras från Suntools. För den som är osäker på hur Suntools startas hänvisas till [13]. Programmet startas från ett kommandofönster, shelltool eller commandtool, med kommandot "visidyn". Skulle datorn inte hitta programmet bör du ta kontakt med den som är systemansvarig och fråga hur du skall gå tillväga.

3.2 Layout

Då programmet startas kommer en panel upp i skärmens överkant. Denna panel kallas huvudpanelen. Här väljer man bl a vilka presentationsformer som ska finnas på skärmen. De fönsterpar som hör till de olika presentations–formerna läggs ut under panelen.

Det finns flera olika sätt att arbeta med fönster. Bl a följande fundamentala strategier kan urskiljas:

- överlappande fönster
- icke överlappande fönster

VISIDYN arbetar med icke överlappande fönster. Användaren hindras från att flytta och ändra storlek på fönstren. I vissa fall är det dock önskvärt att ändra storlek på ett fönster, t ex för att visa Bodediagram där endast en av kurvorna är av intresse. Användaren ska då kunna minska storleken på det fönstret så att enbart den önskade kurvan syns. Detta har möjligtgjorts genom att det finns en knapp i huvudpanelen för att ändra fönsterstorlek. När man trycker på knappen får man upp en ny panel där önskad storlek kan ställas in med ett skjutreglage.

När man anger i huvudpanelen att man vill få upp en ny systempresentation på skärmen får man alltid upp två fönster bredvid varandra. Det vänstra visar det öppna systemet och det högra det slutna. Detta innebär att användaren ibland kan få upp fönster på skärmen som han inte är intresserad av. Anledningen till att vi knyter fönstren så hårt till varandra är främst den, att vi anser att den troligaste användningen av programmet kommer att bestå i att man jämför beteendet hos det öppna respektive slutna systemet när man ändrar ett av systemen. Det underlättar då om fönstren ligger bredvid varandra. Om man sedan vill ändra storlek på ett fönster påverkas båda systemfönstren. Detta innebär att man inte behöver leta efter en viss systempresentation utan det är bara att flytta blicken i sidled. Det finns också en annan fördel med att ha en given placering på fönstren. På det sättet minskar risken att man tar fel och blandar ihop öppna och slutna system.

I appendix C visas en skärmbild som ger en uppfattning om hur det ser ut på skärmen då programmet körs.

3.3 Användning

Inmatning av data till programmet kan ske på två sätt, antingen via musens tre knappar eller från tangentbordet.

Så långt det är möjligt används musen enligt de riktlinjer som Sun har ställt upp för program som använder SunView, dvs enligt följande:

- vänsterknapppen används för att trycka på knappar, markera i rutor och för att välja ut "objekt"
- mittenknappen används för att flytta objekt
- högerknappen används i samband med menyval. Den hålls då ned tills menyn har kommit upp och ett val har gjorts, dvs musen har flyttats tills markören pekar på rätt alternativ

För en noggrannare beskrivning av Suns rekommendationer hänvisas till [11].

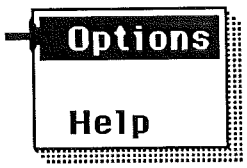
Tangentbordet används enbart för att mata in siffror samt för att ange ett filnamn då man vill spara sitt system på fil, eller hämta ett lagrat system.

I huvudpanelen och i dialogrutor förekommer:

- knappar, t ex "Close", "OK", "Cancel"
- *panelval*, rutor som man markerar i för att få upp en eller flera systempresentationer
- *exklusiva val*, två pilar som bildar en cirkel, används då endast ett alternativ av flera ska gälla
- *inmatningsfält*, fält avsedda för inmatning från tangentbordet, måste väljas ut med **vänster** musknapp innan någonting kan skrivas
- skjutreglage som används för inställning av fönsterstorlek genom att man "tar tag i" det aktuella värdet med **vänster**knappen på musen och justerar, med knappen nedtryckt och utan att lämna reglaget, till önskat värde och därefter släpper man upp knappen

I dialogrutan byter man inmatningsfält med musen genom att klicka på **vänster**knappen i önskat fält eller med **return**- eller **tabulator**tangenten. Avsluta genom att trycka ned **vänster** musknapp i "OK"-rutan. Vill man avbryta inmatning klickar man istället i "Cancel"-rutan. Under tiden som en dialogruta finns på skärmen så är all inmatning omdirigerad så att den går till dialogrutan, dvs innan man kan fortsätta måste man klicka på antingen "Cancel" eller "OK".

De fönster som man kan ändra storlek på har så kallade *fönsterlister* längst ut till vänster. För att se någonting som ligger ovanför fönstret rullar man fönstret genom att klicka på **höger**knappen med markören inom listen. För att rulla åt andra hållet används **vänster**knappen. Genom att klicka på **mitten**knappen i listen hamnar man i en position i fönstret som motsvarar den plats i listen där man klickade.



I varje fönster finns en meny som man får upp om man håller ned den högra musknappen. Val i menyer sker genom att flytta musen tills dess att det rätta alternativet är markerat. Menynalternativet skrivs då med vit text på svart botten (se bild). Sedan släpps musknappen.

Ibland förekommer en liten pil till höger om menyalternativet som visar att det finns ytterligare möjligheter för det menyvalet. Dessa visas som en ny liten meny om man flyttar musen, fortfarande men knappen nedtryckt, så att markören kommer ut till pilen på den aktuella raden. I samtliga menyer i programmet där en sådan här högerpil förekommer gäller att om musknappen släpps upp utan att högermenyn är synlig fås det översta alternativet i högermenyn.

3.4 Uppbyggnad

Programmet är uppbyggt kring en kärna som innehåller följande:

- En panel med gemensamma funktioner, t ex för att spara på fil
- Hantering av fönster och ramar
- Rutiner för att manipulera överföringsfunktioner
- Filhantering
- Rutiner för att visa meddelande- och bekräftelserutor
- Dynamisk listhantering

Alla presentationsformer byggs som separata moduler vilka kopplas till kärnan genom proceduranrop.

Programmet består av cirka 10000 rader C-kod inklusive kommentarer. Dessa rader är fördelade på 11 C-filer och 1 FORTRAN-fil. Här följer en mycket kort beskrivning över C-filerna:

visidyn.c
: Innehåller huvudprogrammet och kod för huvudpanelen, se kapitlet "Huvudpanelen".

com.c
: Denna fil innehåller kärnan, se kapitlet "Kärnan".

visidyn_conf.c
: Här kopplas de olika presentationsmodulerna ihop med kärnan, se kapitlet "Att lägga till egna moduler".

file.c
: Rutiner för filhantering.

list.c
: Källkoden för listhanteringspaketet för dubbellänkade listor, se kapitlet "Att lägga till egna moduler". Till paketet hör en fil med makrodefinitioner, list.h. Det är dessa definitioner som skall användas vis arbete med listor.

message.c
: Denna fil innehåller rutiner för meddelande- och bekräftelserutor.

visidyn_strings.c
: Innehåller strängrutiner som är gemensamma för flera moduler.

bode.c
: Presentationsmodul, se kapitlet "Bodediagram".

singularity.c
: Presentationsmodul, se kapitlet "Singularitetsdiagram".

transfer.c
: Presentationsmodul, se kapitlet "Numerisk överförings–funktion".

select.c
: Innehåller rutiner som behövs i filen transfer.c för att markera texter, se kapitlet "Numerisk överförings–funktion".

C-filerna innehåller sammanlagt cirka 200 procedurer.

För att faktorisera polynom har vi använt en färdigskriven modul, "polynom.f". Denna modul är skriven på institutionen för Reglerteknik vid LTH.

4 Huvudpanelen

Då programmet startas visas huvudpanelen längst upp på skärmen. De knappar och menyval som ingår i panelen beskrivs nedan. Alla operationer som kan skada en överföringsfunktion har försetts med en kontrollfråga.

4.1 Knappval

Resize — Används då man vill ändra storleken på något fönsterpar som innehåller en systempresentation, t ex Bode-fönstret. Programmeraren anger vid installationen av modulen om man skall kunna ändra storlek på ett fönsterpar, och i så fall inom vilka gränser. Det kan därför hända att man får meddelande om att det inte finns något fönsterpar att ändra storlek på, trots att man har ett fönsterpar uppe på skärmen. De numeriska värdena anges i antalet bildpunkter. I sammanhanget kan nämnas att en Sun-skärm har 1152x900 punkter.

Reset — Återställer överföringsfunktionerna till den standard–funktion som finns vid start av programmet.

Store — Sparar de aktuella överföringsfunktionerna på en fil. Inga andra data, såsom vilka fönsterpar som finns på skärmen eller deras storlek, lagras.

Load — Läser in överföringsfunktioner som tidigare lagrats med "Store" (se ovan).

Close — Stäng programmet, dvs visa bara "ikonen". Programmet är nu vilande. När man klickar med vänsterknappen på ikonen öppnas programmet igen. Fönstrens utseende är då det samma som när programmet stängdes.

Quit — Avsluta programkörningen.

4.2 Val av filnamn

Filename: — Här anges den fil med tillhörande biblioteksangivelse som operationerna "Store" och "Load" använder sig av. Då programmet startar finns den aktuella katalogen angiven. Finns den fil man vill använda i aktuell katalog behöver man bara lägga till "/" följt av filnamnet. Stämmer inte katalogen kan det gamla katalognamnet tas bort med DEL-knappen. Därefter kan den korrekta katalogen och filnamnet skrivas in. Innan någonting kan skrivas i detta fält måste man klicka med vänsterknappen i inmatningsfältet.

4.3 Val av presentationsformat

Det finns tre panelval där önskad presentation av överföringsfunktionen väljs, dessa visas nedan. För att få upp ett fönsterpar klickar man med musens vänsterknapp i rutan framför önskad presentation. Fönsterparen uppträder på skärmen i den ordning de väljs. Den inbördes ordningen kan alltså ändras genom att välja och ta bort fönsterpar i olika ordning.

☐ **Singularity diagram**

☐ **Transfer function**

☐ **Bode diagram**

4.4 Menyval

Följande val kan göras i huvudpanelens meny:

Options	Här anges den noggrannhet med vilken numeriska jämförelser skall utföras och om man vill att programmet automatiskt skall förkorta poler och nollställen.
Help	En kortfattad beskrivning av programmet och dess syfte.

4.5 Teknisk beskrivning

Den källkodsfil som innehåller huvudpanelen heter "visidyn.c". Vi kommer här att ta upp de delar som berör kopplingen till och hanteringen av fönsterpar. För en beskrivning av hur man bygger upp en panel hänvisas till [11].

En global lista, repwin_list, innehåller hanterare av typen Repwin_handler för de fönsterpar som är aktiva. En Repwin_handler skapas då ett fönsterpar aktiveras och skickas med till den aktuella modulen vid initieringen av denna. Data en Repwin_handler pekar på:

- Utrymme för länkarna till listan
- Hanterare av typen Window för de båda fönstren.
- En pekare av typen char* på en gömd datastruktur.

Datatypen definieras i "visidyn.h".

Eftersom den ovan beskrivna hanteraren är tillgänglig för de olika modulerna innehåller den en pekare till en gömd datastuktur som endast rutiner som finns i "visidyn.c" och "com.c" kan komma åt. Denna hemliga dataarea är av typen Repwin_client_data*, definieras i "main_panel.h" och innehåller följande:

- En sträng med modulens namn som den skall komma upp i huvudpanelen.
- En unik heltalsidentifierare
- Typen av fönster som skall skapas då presentationsmodulen skall aktiveras t ex CANVAS
- Fönsterparets minimala och maximala höjd. Ett fönsterpar skapas med maximal höjd då det aktiveras. Skulle det inte få plats skapas det så stort som möjligt dock ej mindre än de minimala höjden.
- Pekare till de procedurer som skall anropas då modulen aktiveras, uppdateras och avslutas
- Om modulen är aktiv finns dessutom en Repwin_handler

Denna dataarea kopplas ihop med respektive panelval. Då ett presentationsformat väljs anropas proceduren representation_proc() och Repwin_client_data skickas med som parameter. Detta sköter fönsterhanteringssystemet om. Med hjälp av de uppgifter som finns i denna dataarea skapas ett fönsterpar och en hanterare, Repwin_handler, för detta.

5 Singularitetsdiagram

Panelvalet "Singularity diagram" ger möjlighet att grafiskt placera poler och nollställen i det komplexa talplanet. Genom att klicka i rutan framför texten "Singularity diagram" får man upp två fönster. Det vänstra fönstret visar det öppna systemet och det högra det slutna. I de två fönstren är det sedan möjligt att med enkla medel "bygga upp" det system man vill undersöka. All interaktion med användaren sker via musen med vars hjälp man väljer i menyn, placerar ut, flyttar samt markerar singulariteter.

När man har fått upp fönstren på skärmen kan man börja arbetet med att placera ut singulariteterna där man vill ha dem. Singulariteterna kan enbart placeras ut innanför den "ram" som i vänstra och undre kanten tjänstgör som skala. För att inte skalan ska bli alltför svårläst skrivs endast den övre och undre gränsen ut. Om de inställda gränserna medger det ritas koordinataxlarna ut för att man lättare ska kunna orientera sig .

5.1 Menyalternativ

I de två fönstren finns menyer med följande alternativ:

New zero	skapar ett nytt nollställe
New pole	skapar en ny pol
Delete	tar bort en singularitet
Move	flyttar en singularitet
Set scales	ändrar det område som visas inom fönstrets ram
Help	beskriver av vad som kan göras i fönstret

För de menyval som har en högermeny gäller att menyvalen i huvudmenyn är de som oftast kommer att användas. I högermenyerna finns det möjlighet att avgöra om det man vill utföra ska gälla en eller flera singulariteter. Som standard har vi valt att man jobbar med en pol eller ett nollställe i taget.

5.2 Att skapa nya singulariteter

För att skapa en ny singularitet väljer man det av alternativen i menyn som svarar mot den typ av singularitet som man vill ha. Markören ändras då från att ha varit en pilspets till att bli ett hårkors, detta för att man lättare ska kunna välja rätt position för polen eller nollstället. Som ytterligare hjälp vid utplaceringen visas den aktuella real- och imaginärdelen ovanför den ram som begränsar arbetsytan. Dessa koordinater visas för övrigt inte bara när man skapar en singularitet utan så fort någon är "utvald". När musen har flyttats så att hårkorset befinner sig på rätt ställe trycks **vänster**knappen ned. Därmed har singulariteten placerats ut. På den plats där en pol eller ett nollställe placerats ritas en liten figur som anger position och typ för singulariteten.

Om singulariteten placeras på ramen runt arbetsområdet får man en singularitet som inte är markerbar. Vill man kunna markera den måste skalan ändras. Detta är samma typ av singulariteter som de som hamnar utanför det visade området, t ex på grund av skaländring eller omräkning mellan öppna och

slutna systemet. Singulariteterna läggs på ramen för att visa att de ligger utanför det visade området

Skulle man innan utplaceringen är klar komma på att man håller på med något galet kan man avbryta utplaceringen genom att flytta ut markören utanför det aktuella fönstret, eller trycka ned **vänster**knappen utanför ramen som visar arbetsområdet. Det är ingen katastrof om man av misstag skulle komma utanför ramen, utan att lämna fönstret, eftersom hårkorsmarkören återkommer så fort man kommer innanför ramen igen och utplaceringen kan fortsätta som innan.

Det tål att påpekas att man kan placera ut singulariteter både i det öppna och i det slutna systemet. Man kan alltså börja med att välja beteende för det återkopplade systemet och se efter hur motsvarande öppna system skulle se ut. Denna flexibilitet är ju den stora förtjänsten med VISIDYN.

Förutom den rent grafiska utplaceringen av singulariteter finns det en möjlighet att via högermenyerna till "New zero" och "New pole" ange singulariteten med siffror. Denna facilitet är närmast tänkt att användas då det inte finns tillräckligt med plats på skärmen för överföringsfunktionsfönstren. Om det finns någon singularitet markerad då detta alternativ väljs fås som förslag de koordinater den markerade singulariteten har, i annat fall föreslår programmet att singulariteten ska ligga i origo.

Om det finns multipla singulariteter på en viss koordinat kommer en speciell bild att ritas. Denna bild är densamma som den vanliga bilden med undantag för att den är något "fetare".

5.3 Markering av singulariteter

Då en singularitet skapats, ritas den på skärmen i inverterat tillstånd, vi säger att singulariteten är markerad. En markerad singularitet intar en särställning i programmet. Det är endast markerade singulariteter som kan flyttas och tas bort. Man kan naturligtvis markera en annan singularitet än den senast skapade, förutsatt att det finns flera singulariteter. Detta sker genom att musen flyttas tills pilspetsmarkören pekar på den aktuella singulariteten, varefter man klickar med den **vänstra** musknappen. Om det finns en markerad singularitet står dess koordinater angivna ovanför ramen. Om man klickar med vänsterknappen någonstans inom arbetsytan där det inte finns någon singularitet säger vi att man gör en avmarkering. Om det då fanns en pol eller ett nollställe markerat ritas singulariteten om så att den antar sitt normala utseende.

Om man markerar en multipelsingularitet ritas en "fetare" version av den inverterade bilden. Skulle det ligga både poler och nollställen på samma ställe så innebär markering det samma som alternering mellan pol och nollställe.

På grund av att bilderna av singulariteterna består av bitmönster med endast 7x7 punkter blir den inverterade bilden ibland något otydlig, speciellt då flera singulariteter ligger väldigt nära varandra.

5.4 Borttagning av singulariteter

När en singularitet är markerad kan man ta bort den genom att välja "Delete" i menyn. Om det skulle finnas flera poler eller nollställen på samma plats som den markerade kommer endast en att tas bort om man väljer det normala "Delete". Vill man ta bort mer än en singularitet vid ett visst tillfälle kan man

använda alternativet "Delete poles/zeros" som finns i högermenyn till "Delete".

Om det inte finns någon markerad singularitet i ett system så är "Delete"-alternativet i motsvarande fönsters meny inaktivt och kan ej väljas.

5.5 Att flytta singulariteter

Med menyvalet "Move poles/zeros" i högermenyn till "Move" kan en eller flera singulariteter flyttas genom att de nya koordinaterna samt antalet singulariteter anges.

För att flytta **en** singularitet finns det en genväg så att man inte behöver använda menyn. Flytta musen tills markören pekar ut rätt singularitet, tryck på och håll ned, **mitten**knappen på musen. Markören ändras då till den inverterade bilden på singulariteten och hjälpkoordinaterna ovanför ramen kommer fram. När markören har flyttats till önskad position släpper man musknappen och förflyttningen är utförd.

Finns det inte någon singularitet markerad går det inte att flytta singulariteter via menyn.

5.6 Ändring av arbetsytans storlek

Menyalternativet "Set scales" används då man vill ändra det område som visas inom ram i fönstret. Man får upp en dialogruta där min- och maxvärde både för reella och imaginära axeln kan anges. Med detta alternativ kan man välja att enbart titta på övre halvplanet genom att sätta den undre gränsen för imaginärdelen till noll. Detta medför att man för en given övre gräns kan nå en större upplösning. Ett tips är dock att ta med en liten del av det undre halvplanet, annars kommer alla reella singulariteter att ligga på ramen och därmed är de ej valbara.

5.7 Diskussioner

När singulariteterna placeras ut räknas positionen ut med en precision av en bildpunkt. Vi har alltså valt att inte "klumpa" ihop bildpunkterna för att på så sätt få en viss indelning utav skärmen utan noggrannheten i utplacerandet avgörs av aktuell skala. Här skulle man naturligtvis ha kunnat tänka sig att om positionen hamnade inom en ruta, låt säga 3x3 punkter stor, så hade singulariteten lagts mitt i rutan. Storleken på rutan skulle då ha valts via ett menyalternativ men detta ansågs inte tillföra programmet någonting nytt, utan vi valde att avstå från denna finess.

En annan finess som vi valde att inte införa i programmet är möjligheten att få multipliciteten för en eller samtliga singulariteter utskriven bredvid objektet. Anledningen till att detta inte infördes var att det i vissa situationer skulle kunna bli trångt på skärmen, t ex om skalan är vald så att det skiljer väldigt mycket mellan övre och undre gränserna. En alternativ lösning skulle kunna vara att man har ett menyalternativ för att få multipliciteten för den **markerade** singulariteten utskriven exempelvis bredvid koordinaterna ovanför ramen, men inte heller detta är infört.

Ett tag diskuterade vi om man enbart skulle visa övre halvplanet och att alla komplexkonjugerade singulariteter skulle vara underförstådda men vi valde att låta användaren själv avgöra detta med hjälp av skalinställningen.

Vi har valt att ha en fix storlek på fönstren för singularitetsdiagrammen, detta med tanke på att fönsterlister inkräktar på ritytan på ett något konstigt sätt. Detta hade dock gått att ordna men eftersom det endast hade betytt mer arbete så valde vi att avstå. Vill man tvunget ändra storlek på fönstret hänvisas till den del av den tekniska beskrivningen som behandlar nya moduler.

En skillnad mellan singularitetsfönstret och överföringsfunktionsfönstret, se kapitlet "Numerisk överföringsfunktion" är att man inte kan ändra förstärkningen i singularitetsfönstret. Om man på ett enkelt sätt hade kunnat variera förstärkningen skulle man få rotorter "på köpet". För ett förslag på hur man enkelt skulle kunna införa ändring av förstärkningen hänvisas till avsnittet som beskriver singularitetsmodulens uppbyggnad.

I en del sammanhang är den relativa dämpningen och frekvensen för en singularitet intressantare att känna till än koordinaterna. Det skulle alltså vara trevligt att kunna läsa av dämpning och frekvens för en viss singularitet. Man skulle också kunna utvidga funktionen med inmatningsmöjligheter.

5.8 Teknisk beskrivning

Varje singularitet representeras i "singularity.c" av en post innehållande

- typ av singularitet
- multiplicitet
- vilket system singulariteten finns i
- en logisk variabel som anger om singulariteten ligger innanför den ram som anger arbetsområdet
- pekare på de bilder som visar den normala singulariteten, den inverterade singulariteten samt multipla singulariteter
- en vektor med två element som är poster innehållande singularitetens komplexa koordinater, koordinater i bildpunkter, "pixel", för den bild som motsvarar singulariteten samt en systemdefinierad datastruktur, "Rect", som används för att kontrollera om en musknapp tryckts ned inom singularitetens bild. För en beskrivning av typen Rect hänvisas till [9]. Om singulariteten är reell används bara det andra av de två elementen, i annat fall, dvs singulariteten är komplexkonjugerad, används båda.

Den verkliga hörnstenen i modulen är proceduren "singularity_event_proc" som anropas av det underliggande fönsterhanteringssystemet så fort musens tillstånd har ändrats, t ex då musen har flyttats eller någon knapp har tryckts ned. Rutinen använder ett antal globala variabler som dessvärre är nödvändiga eftersom programmet är händelsestyrt. Variablerna används för att hålla reda på om en ny singularitet är under utplacerande, om en singularitet flyttas etc.

För att sköta om all uppdatering i fönstret på order från kärnan finns det en procedur "update_singularity_win". Denna är uppbyggd enligt samma mall som motsvarande uppdateringsrutiner i de andra modulerna.

I avsnittet som beskriver vad som kan göras respektive inte göras i singularitetsfönstren tar vi upp en utvidgning av programmet för att ändra förstärkning på ett enkelt sätt, och för att på så sätt införa rotorter. Det vi

tänkte på då var att man kan införa två bilder, en uppåtriktad och en nedåtriktad pil, kopplade till varsin "Rect" samt en ruta där värdet skrivs ut. Man kan sedan känna av om en viss musknapp, lämpligen den vänstra, tryckts ned inom pilen och i så fall skicka en uppdateringsorder till kärnan. Det man måste fundera över är hur stora steg förstärkningen ska ändras med, detta bör man lämpligen kunna ändra via ett menyval, samt vad som ska hända om musknappen hålls ned under en längre period. Här måste man troligen lägga in en liten timer för att med jämna mellanrum skicka uppdateringsorder till kärnan. Ett alternativ, som är mer likt suntools fönsterlister, är att endast ha en pilruta och använda både vänster och höger musknapp för att ändra förstärkningen.

6 Numerisk överföringsfunktion

Med panelvalet "Transfer function" får man en numerisk presentation av systemet där man kan lägga till, ta bort och flytta singulariteter. Det vänstra av de två fönstren visar det öppna systemet medan det högra visar det slutna.

Det som visas i fönstret under rubrikerna "Zeros" och "Poles" är överföringsfunktionen med A- och B-polynomen uppdelade i sina faktorer. Dessutom visas systemets förstärkningsfaktor under rubriken "Const".

6.1 Menyalternativ

I varje fönster finns en meny och i det här fallet finns följande alternativ i menyn:

New zero	skapar ett nytt nollställe
New pole	skapar en ny pol
Delete pole(s)/zero(s)	tar bort en eller flera singulariteter
Move pole/zero	flyttar en singularitet
Change constant	ändrar systemets förstärkning
Show polynomials	visar överföringsfunktionen
Help	ger en kort hjälptext för fönstret

6.2 Användning

För att lägga till en ny singularitet i ett av systemen väljer man "New zero" eller "New pole" i den meny som finns i det fönster som svarar mot det rätta systemet. I den dialogruta man då får upp skrivs koordinater och multiplicitet. Singulariteterna visas därefter på skärmen på formen x±iy. Om x eller y är noll skrivs de ej ut. Är däremot både x och y noll skrivs singulariteten som "0". Finns mer än en singularitet på en viss koordinat omsluts koordinaterna med parenteser och sedan hängs "^multiplicitet" på efter.

När en singularitet skapats kan den väljas ut (markeras). Man flyttar då musen tills pilspets-markören pekar ut den singularitet man vill markera och trycker ned **vänster**knappen på musen. Om det finns en pol eller ett nollställe markerad när man väljer något av "New"-alternativen i menyn så kommer den markerade singularitetens koordinater, oavsett vilken typ av singularitet som är markerad, att stå i dialogrutan och om man inte anger något annat så fås de värdena. Om man klickar med musens vänsterknapp på någon plats i fönstret där det inte står någon text innebär det att man avmarkerar eventuellt markerad singularitet. Det kan ibland vara en fördel då man ska skapa en ny singularitet och vill nollställa standardvärdena.

För att ta bort en markerad singularitet väljer man "Delete pole(s)/zero(s)" i menyn. Man får då upp en dialogruta som frågar om det är OK att ta bort singulariteten.

När man vill flytta en singularitet kan man gå till väga på två olika sätt. Antingen väljer man "Move pole/zero" i menyn eller också klickar man med

mittenknappen på den aktuella singulariteten. Det första alternativet kräver att en pol eller ett nollställe är markerat medan det andra är en kombination av en markering och ett menyval.

Om man vill titta på A- och B-polynomen väljer man "Show polynomials" i menyn, och får då upp en meddelanderuta. I denna ruta kan man läsa av polynomen men **ej** ändra dem.

6.3 Diskussion

Vi hade en del funderingar på att tillåta inmatning av polynomen men det visade sig att det helt saknades stöd i SunView för att göra kontrollerade ändringar mitt inne i ett inmatningsfält så vi valde att avstå från denna funktion.

Liksom i singularitetsdiagrammet skulle det vara intressant att kunna läsa av och eventuellt ändra den relativa dämpningen och frekvensen för en singularitet.

6.4 Teknisk beskrivning

Den interna datastrukturen i "transfer.c" är väldigt enkel beroende på att grovjobbet görs i "select.c". Följdaktligen är det där som datastrukturen blir komplicerad. För att hålla reda på singulariteterna sparas poster innehållande följande data i en 2x2 matris av listor:

- typ av singularitet
- vilket system singulariteten finns i
- singularitetens komplexa koordinater, för imaginärdelen lagras det positiva värdet om det är en komplexkonjugerad singularitet
- multiplicitet
- rad och kolumn för första tecknet i den teckensträng som skrivs i fönstret, rad och kolumn räknas i tecken och inte i bildpunkter
- teckensträng

Detta var den första modul som skrevs. Det är möjligt att datastrukturen skulle kunna förenklas något med den erfarenhet som vi fick efter att ha skrivit ytterligare två moduler.

Den sträng som en viss pol eller ett visst nollställe ger upphov till placeras ut i fönstret efter en viss strategi. Vi försöker så långt som det är möjligt få snygga spalter med en fix bredd. Om en sträng skulle vara längre än denna bredd används den dubbla spaltbredden. Endast enkel och dubbel spaltbredd förekommer. På så sätt undviker man den "röra" som maximal packning av strängarna skulle medföra. Det förenklar arbetet med att räkna ut var nästa teckensträng ska placeras om man har fyra listor med singulariteter. Därför är singulariteterna ordnade i en matris där första index i matrisen anger vilket system det rör sig om och andra index anger singularitetstyp. Ett särskilt problem som man måste se upp med är den "rullning" av raderna som kan uppstå då en singularitet tas bort eller då ett nollställe läggs till. I det sista fallet berör rullningen endast raderna med poler.

För att sköta om all uppdatering i fönstret på order från kärnan finns det en procedur "update_transfer_win". Denna är uppbyggd enligt samma mall som motsvarande uppdateringsrutiner i de andra modulerna.

Den händelserutin som finns i "transfer.c" hanterar enbart högerknappen, dvs menyval. Detta beror på att all hantering och utmärkning av strängar sker i "select.c". Det görs på så sätt att man till det aktuella fönstret knyter en post, som innehåller allt som behövs för att hålla reda på samtliga valbara och ej valbara strängar som skrivs i fönstret. I denna post ingår:

- pekare på ritytan och på det typsnitt som används för att skriva strängarna
- aktuell rad och kolumn, i tecken
- maximalt antal rader och kolumner i fönstret
- vissa data gällande typsnittet
- en lista över samtliga valbara objekt
- en pekare på aktuellt objekt
- en pekare på det objekt som eventuellt håller på att definieras
- en lista med "tomma" definitioner för att minska minnesfragmenteringen
- en pekare på den procedur som ska ta hand om de händelser som inte berör val eller avmarkering av objekt, t ex menyhantering
- en pekare på en procedur som ska anropas då avmarkering av ett objekt sker

Varje singularitet representeras av en post som länkas in i listan över valbara objekt. Denna post innehåller det som är specifikt för objektet ifråga:

- vilken rad teckensträngen står på
- start- och slutkolumn för strängen
- strängen
- en pekare till objektets data, dvs i det här fallet koordinater, singularitetstyp och multiplicitet
- en pekare på en funktion som kan jämföra två olika objekts data
- pekare på två funktioner som ska anropas då man klickar på objektet med musens vänstra respektive mittersta knappar
- en länk till nästa post i listan

Den här posten får sina värden då objektet skapas, dvs efter ett anrop från den funktion i "transfer.c" som hanterar all uppdatering av fönster.

Det förtjänar att påpekas att "select.c" är skriven för att klara en allmän tillämpning där man hanterar strängar i fönster. Det motiverar pekaren till en jämförelsefunktion.

7 Bodediagram

Ett *Bodediagram* visar på ett lättfattligt sätt egenskaperna hos en överföringsfunktion vid olika frekvenser. Det är t ex mycket enkelt att bestämma fas- och amplitudmarginalerna ur ett Bodediagram.

7.1 Teoretisk bakgrund

Antag att vi har följande enkla överföringsfunktion:

$$G(s) = \frac{a}{a+s} = \frac{1}{1+s/a}$$

Sätt $s = i\omega$ och beräkna beloppet och fasen av resultatet:

$$|G(i\omega)| = \frac{1}{\sqrt{1+(\omega/a)^2}}$$

$$\arg G(i\omega) = -\arg(1+\frac{i\omega}{a}) = -\arctan\frac{\omega}{a}$$

Beloppet av överföringsfunktionen skall plottas i *logaritmisk* skala. Logaritmering ger:

$$\log|G(i\omega)| = -\frac{1}{2}\log(1+\frac{\omega^2}{a^2})$$

där log betyder 10-logaritmen.

I stället för att plotta kurvorna kan man rita asymptoter. Då man ritar för hand är detta en metod att snabbt skissa en belopp- eller amplitudkurva. Då det gäller beloppkurvan ser man lätt att för små värden på vinkelfrekvensen ($<<a$) blir det logaritmerade värdet approximativt lika med noll vilket motsvaras av en vågrät linje i diagrammet. Om vi däremot ansätter ett stort värde på vinkelfrekvensen ($>>a$) får man en linje med lutningen -1. Skärningen mellan de båda linjerna sker i punkten a vilken kallas *brytpunkt*.

Att få fram asymptoterna till faskurvan är något besvärligare. Vi antog att faskurvans lutning var störst vid brytfrekvensen. Genom att räkna ut lutningen i denna punkt får man en linje som approximerar övergången mellan kurvans lågfrekvensasymptot, som är en vågrät linje i nollan, och högfrekvensasymptoten, som är en vågrät linje i $-\pi/2$.

Lutningen vid brytfrekvensen beräknas på följande sätt:

$$\frac{d}{d\log\omega}(-\arctan\frac{\omega}{a}) = \ldots = \ln(10)\frac{\omega}{a}\frac{1}{1+(\omega/a)^2} = [\omega = a] = -\frac{\ln(10)}{2}$$

Vi ser att lutningen som väntat är helt oberoende av vinkelfrekvensen.

Samma resonemang som ovan kan tillämpas på nollställen varför vi inte kommer att beröra dessa alls här. Vi skall i stället titta på vad som händer om man har en komplexkonjugerad pol. Överföringsfunktionen kan då skrivas:

$$G(s) = \frac{\omega_0^2}{s^2+2\zeta\omega_0^2 s+\omega_0^2} = \frac{1}{\frac{s^2}{\omega_0^2}+2\zeta\frac{s}{\omega_0}+1} \qquad \text{där } \zeta < 1$$

Fas- och beloppkurvorna beräknas på samma sätt som i fallet med en enkel pol. Det enda som kan vålla problem är beräkningen av faskurvans asymptoter. Vi går därför snabbt igenom hur man kan gå tillväga.

Precis som i fallet med enkelpolen består faskurvan av en lågfrekvensasymptot, vilken är en vågrät linje genom nollan, och en högfrekvensasymptoten, som på samma sätt är en vågrät linje i $-\pi$.

Lutningen vid brytfrekvensen beräknas precis som tidigare:

$$\frac{d}{d\log\omega}\left(-\arctan\frac{2\zeta\omega\omega_0}{\omega_0^2-\omega^2}\right) = \ldots = -\ln(10)\frac{\omega}{\zeta\omega_0} = [\omega=\omega_0] = -\frac{\ln(10)}{\zeta}$$

Vi får alltså en lutning som blir brantare ju mindre ζ blir dvs ju närmare den reella axeln polerna ligger.

Genom att kurvorna tack vare logaritmeringen kan adderas kan man med hjälp av de ovan beskrivna komponenterna bygga upp en kurva som beskriver en godtycklig överföringsfunktion.

7.2 Användning

I fönstret finns två diagram. Det övre visar beloppkurvans asymptoter medan det undre visar asymptoterna till faskurvan. Då en pol eller ett nollställe markeras i något annat fönster (det går **ej** att markera i Bodediagrammet) markeras singularitetens läge i beloppkurvan med ett kryss och de numeriska värdena visas på raden mellan de båda diagrammen. Observera att medan kurvorna ritas med hjälp av asymptoter, anges de numeriska värdena för enskilda punkter exakt.

Bode-fönstren skiljer sig från de båda andra presentationsformerna så tillvida att det **inte** går att förändra en överföringsfunktion. Det går alltså inte att lägga till poler och nollställen eller att ändra förstärkningen.

För att kunna arbeta med Bodefönstren måste man veta vad de olika knapparna på musen används till. Här nedan följer en kort beskrivning:

Vänster Används ej.

Mitten — Denna knapp används för mätningar i diagrammen. Det går till så att man placerar pilmarkören i något av diagrammen och trycker ned mittenknappen. Markören kommer då ett ersättas av två lodräta steck, ett i vardera diagrammet. Genom att föra musen i sidled kan mätpunkten flyttas. De numeriska värden som gäller för mätpunkten anges ovanför beloppkurvan. Observera att medan kurvorna ritas med hjälp av asymptoter anges de numeriska värdena för mätpunkterna exakt. Med mätfunktionen kan man enkelt bestämma amplitud- och fasmarginal.

Höger — Används i menyval på vanligt sätt.

Följande menyval finns:

Set scales — Här anger man vilka skalor det skall vara i diagrammen. Observera att vinkelfrekvensen anges gemensamt för de båda diagrammen.

Help — Genom att välja detta menyalternativ får man upp ett fönster innehållande en kort beskrivning av vad som visas i fönterna och vilka funktioner som finns tillgängliga.

De funktioner som beskrivits ovan är valda så att man på ett lätt och naturligt sätt skall få fram den väsentliga informationen ur de båda diagrammen.

7.3 Teknisk beskrivning

Alla poler och nollställen lagras i två länkade listor, en för vardera systemet. Dessa ligger i sin tur i den globala strukturen "systems". För varje pol/nollställe lagras förutom den beskrivning som kommer från kärnan också information om brytfrekvenser, dämpning och belopp.

Det finns ytterligare en stor global struktur som heter "scales". Denna innehåller skalorna för de olika diagrammen.

Precis som i alla andra moduler finns det två procedurer som är centrala:

bode_event_proc()
Denna procedur tar hand om all information från musen då den befinner sig i Bodefönstret.

update_bode_win()
Kärnan anropar denna procedur ett antal gånger för varje ändring av överföringsfunktionerna.

Övriga delar beräknar fas- och beloppkurvor, hanterar menyn där man anger skalor på axlarna etc.

7.4 Synpunkter

Det första man lägger märke till är att det bara är asymptoterna till kurvorna som ritas. Detta kan upplevas som otillräckligt speciellt då överföringsfunktionen innehåller komplexkonjugerade poler/nollställen och därmed resonanstoppar. Anledningen till att endast asymptoterna ritas är tiden. Programmet skall reagera utan märkbar fördröjning på varje ändring av överföringsfunktionerna. Detta hade inte varit möjligt om de verkliga kurvorna skulle ritats ut. Vi har därför kompromissat så tillvida att visserligen ritas bara asymptoterna ut men alla mätningar och lägesangivelser för

poler/nollställen anges exakt. Den som vill få en bättre uppfattning om hur kurvan ser ut kan alltså använda sig av mätfunktionen (se ovan).

Markering av poler/nollställen är något som skulle varit möjligt att ha med. Dock endast i beloppdiagrammet eftersom det är svårare att på ett naturligt sätt definiera ett exakt läge för en singularitet i fasdiagrammet. Denna möjlighet bedömdes dock inte allför väsentlig att implementera eftersom Bodekurvorna i de flesta fall är ett komplement till de andra presentationsformerna. Markeras en pol/nollställe i ett annat fönsterpar kommer dess läge i Bode–diagrammet att markeras. Detta borde vara tillräckligt i de allra flesta fall.

Att förändra en överföringsfunktion genom att ändra kurvornas utseende skulle vara en mycket trevlig finess. Det för dock med sig vissa praktiska svårigheter. Antag att man vill lägga till en dubbelpol. Denna definieras dels av en brytfrekvens, dels av en dämpning. Brytfrekvensen kan lätt definieras genom att föra markören i sidled längst frekvensaxeln. Frågan är hur man bäst definierar dämpningen. Så vitt vi kan se finns det två vägar att gå. Den ena går ut på att man anger resonanstoppens storlek. Den andra metoden är att ange en fasvridning för den aktuella brytfrekvensen. Båda dessa metoder har för och nackdelar. Den första metoden har den fördelen att man anger båda parametrarna i samma diagram. Nackdelen är att resonanstoppens storlek knappast är av primärt intresse vid designen av en överföringsfunktion. Då är fasvridningen betydligt intressantare vilket för oss in på den stora fördelen med den andra metoden. Nackdelen här är att det inte känns riktigt naturligt att ange en polplacering i fasdiagrammet. Ännu värre blir det om man bara skall flytta en befintlig pol. En lösning på detta problem är att rita både fas- och beloppkurvan i samma diagram. Detta tenderar dock att bli ganska rörigt.

8 Kärnan

Den centrala delen av programmet, här kallad kärnan, är den som tar emot alla ändringar av överföringsfunktionerna. Här beräknas sedan de nya överföringsfunktionerna för det öppna och slutna systemet. Då detta är klart skickas meddelande till alla aktiva moduler om vilka poler och nollställen som har ändrats. Hela denna procedur måste utföras så snabbt att användaren inte märker någon nämnvärd fördröjning i svarstiderna.

Kapitlet "Att lägga till egna moduler" beskriver hur man använder kärnan och skriver egna moduler.

8.1 Representation av överföringsfunktionerna

Överföringsfunktionerna representeras av två moniska polynom, A(s) och B(s), och en förstärkningsfaktor, K. Vi har valt att representera de båda polynomen på faktorform. Överföringsfunktionen representeras på faktorform enligt

$$G(s) = K\frac{B(s)}{A(s)} = K\frac{(s-z_1)\ \ldots\ (s-z_m)}{(s-p_1)\ \ldots\ (s-p_n)}$$

Detta val har gjorts främst av den anledningen att då en överföringsfunktion skall ändras berörs i de flesta fall endast en eller ett fåtal av polerna/nollställena. Detta gör att de uppdateringar som görs av samtliga fönster bara behöver beröra delar av överföringsfunktionerna. Dessutom behövs singulariteternas placering för de flesta vanliga grafiska presentationer av en överföringsfunktion. Det är därför naturligt att arbeta med polynomen på faktoriserad form. En representation på faktorform är också bättre ur numerisk synpunkt.

Nu är det tyvärr så att då man omvandlar från ett öppet system till ett slutet använder man addition och subtraktion, se ekvation (1.1) och (1.2). De båda senare operationerna är svåra att göra på polynom i faktoriserad form och det får till följd att en omräkning till "vanlig" polynomform är nödvändig för dessa operationer. Dessa polynom måste sedan faktoriseras vilket är en "flaskhals" i systemet som begränsar antalet poler och nollställen i överföringsfunktionen. Dessutom förlorar man i numerisk noggrannhet.

8.2 Diskussion

En fråga som vållade lite huvudbry är huruvida en singularitet skall kunna låsas till att alltid ligga i en viss position. För att belysa varför man vill ha denna facilitet kan vi ta ett enkelt exempel:

Ex.

> En process skall regleras. Denna process har ett antal poler och nollställen som markeras i det öppna systemet. Dessa singulariteter är fasta och kan inte ändras. Dessutom tillkommer ett antal poler och nollställen som tillhör regulatorn. Dessa skall justeras så att önskade egenskaper erhålles i det slutna systemet. Det skulle då vara önskvärt att kunna gå in i det slutna systemet och ändra på detta till önskat utseende utan att processens poler och nollställen i det öppna systemet berördes. All justering skulle alltså ske med regulatorns poler och nollställen.

Detta kan vara ett besvärligt problem som inte alltid har en lösning. Det kan också hända att man måste flytta på flera poler/nollställen i det slutna systemet för att de "låsta" singulariteterna i det öppna systemet skall behålla sina platser. Följden blir att om man flyttar en pol i det slutna systemet så kommer ett antal andra singulariteter också att flytta på sig. Detta kan upplevas som förvirrade. Vi har inte utrett villkoren för att en låsning skall vara möjlig. Intuitivt kan man säga att om det finns många frihetsgrader, dvs många singulariteter i regulatorn, så är det troligare att man kan hitta en lösning.

8.3 Teknisk beskrivning

Det finns i kärnan två globala variabler, "systems" som innehåller överföringsfunktionerna och "changed_in_systems" som innehåller de ändringar som gjorts i överföringsfunktionerna.

Systembeskrivningen i "systems" är av typen System_type och innehåller:

- Information som anger om någon singularitet är markerad och i så fall vilken.
- En vektor med två element, ett för det öppna och ett för det slutna systemet. Dessa element är av typen Transfer_function.

En beskrivning av en överföringsfunktion lagras i en post av typen Transfer_function. Följande ingår:

- Förstärkningsfaktorn K
- En vektor innehållande koordinaterna för alla singulariteter i överföringsfunktionen och en angivelse av hur många poler respektive nollställen det finns i denna position. Dessutom finns en markering i varje element som talar om ifall elementet utnyttjas. Storleken på vektorn utgör en begränsning på gradtalet i överföringsfunktionen. I filen "com.h" finns konstanten MAX_POWER som anger det största tillåtna gradtalet.
- Ett heltal som anger maximalt använda index i vektorn

Att vi har valt att lägga alla väden i en vektor i stället för att använda en länkad lista beror på effektiviteten. Det tar längre tid att reservera minne för varje ny koordinat än att lägga in den i en vektor. Det tas också en kopia av överföringsfunktionerna innan nya ändringar införs. Detta för att ha en korrekt variant kvar om något skulle misslyckas under uppdateringen.

Kopieringen blir effektivare genom en direkt posttilldelning av hela systemet än om varje element skulle kopieras var för sig.

De flesta av de i modulen ingående funktionerna finns beskrivna i kapitlet "Att lägga till egna moduler" varför dessa inte kommer att gås igenom närmare här. Det kan nämnas att för att hitta rötterna till polynomen använder vi en färdigskriven FORTRAN-rutin vilken ursprungligen skrevs för att användas på VAX-datorer. Vi har modifierat denna rutin så att den går under UNIX på Sun och dessutom arbetar i dubbel precision.

Eftersom meningen med programmet är att användaren skall kunna ändra något i en överföringsfunktion, och omedelbart få det motsvarande öppna eller slutna systemet, ställs stora krav på snabbhet i den del som gör det tunga arbetet. Vi har lagt ner en hel del tid på att få kärnan att arbeta effektivt. Ett exempel på optimering är att alla uppdateringar skickas till en modul i taget i stället för att en ändring i taget skickas till alla. Detta undviker en hel del "swapping" mot skivminnet.

9 Att lägga till egna moduler

Det finns många sätt att grafiskt presentera en överföringsfunktion. De metoder som följer med programmet är de mest grundläggande. Det kan finnas tillämpningar där de inte räcker till. Skulle så vara fallet är det enkelt att lägga till nya moduler. Här följer en beskrivning av hur en modul skall vara uppbyggd och hur denna sedan kopplas ihop med kärnan. I detta kapitel förutsätts att läsaren har relativt goda kunskaper om C-programmering i UNIX-miljö. För den som är osäker på programspråket C hänvisas till [5].

9.1 Globala datatyper och konstanter

Alla moduler som skall kopplas samman med VISIDYN skall göra "include" på filen "visidyn.h". I denna fil definieras de konstanter och datatyper som är gemensamma för kärnan och de olika modulerna. Vi tar här bara upp de viktigaste datatypernas användning.

Repwin_handler
: Detta är en hanterare som är unik för varje fönsterpar. Den skapas av kärnan då en modul aktiveras och skickas med vid initieringsanropet till modulen. Den används sedan för att identifiera modulen vid vissa anrop till kärnan. De viktigaste är kanske att hämta hanterarna av typen Window för de båda fönstren. Detta görs genom anropen open_loop_win(rh) och closed_loop_win(rh), där rh är en variabel av typen Repwin_handler.

Complex
: Används för att representera komplexa tal. Det är en struktur innehållande två komponenter, realdel re och imaginärdel im. Dessa är av typen double.

Count_ptr
: Detta är en pekare till en struktur som innehåller en heltalsvektor. Vektorn har två element, p->n[ZERO] och p->n[POLE] och används för att ange antalet poler och nollställen vid uppdatering av de olika modulerna.

Polynomial
: Innehåller ett polynom med reella koefficienter. För närmare beskrivning se filen "com.h".

Det finns ytterligare några datatyper och ett antal konstanter som man bör känna till. För närmare beskrivning av dessa hänvisas till filen "visidyn.h".

9.2 Presentationsmodulers uppbyggnad

All hantering för att skapa, ta bort och ändra storlek på fönster sköts av kärnan. En modul måste veta när den är aktiv, dvs när den är vald i huvudpanelen. Därför skall det i varje modul finnas en rutin som anropas då modulen aktiveras och fönstren kommer upp på skärmen. På samma sätt skall det finnas en rutin som anropas precis innan fönstren tas bort för att modulen skall kunna städa efter sig. Dessutom skall det finnas en rutin som anropas varje gång överföringsfunktionerna har ändrats. Denna rutin sköter all

uppdatering av fönstret. Genom detta uppnår man att de olika presentationsfönstren alltid är aktuella. Det finns dock ett tillfälle då man kan tillåta att en annan rutin uppdaterar skärmen. Det är när man känner av musrörelser med **nedtryckt** musknapp. Anledningen är att det inte ska skickas uppdateringsorder till de andra fönstren vid varje liten förflyttning utan först då knappen släpps upp.

Här följer en kort beskriving av procedurerna för att skapa, uppdatera och ta bort fönster. Procedurnamnen kan väljas fritt men måste vara unika. Namnen på procedurerna anges i det proceduranrop som görs i filen "visidyn_conf.c". Denna fil kopplar ihop modulerna med kärnan, för vidare information se avsnittet "Installering av ny modul" i detta kapitel. Det rekommenderas att man följer den namnkonvention som anges.

```
create_<modulnamn>_win(rh)
Repwin_handler rh;
```

Denna rutin anropas omedelbart efter det att de båda fönstren har skapats. Argumentet är en datatyp som bl a innehåller hanterarna av typen Window för de båda fönstren. Dessa behövs då man skall rita eller skriva, och man kommer åt dem genom de båda macro-anropen open_loop_win(rh) och closed_loop_win(rh). Efter anropet av denna funktion skickar kärnan de uppdateringsmeddelanden som krävs för att den aktiverade modulen skall erhålla de aktuella överföringsfunktionerna.

```
delete_<modulnamn>_win()
```

Anropas omedelbart innan fönstren tas bort. Detta för att modulen skall ha en möjlighet att städa efter sig. Det kan t ex gälla avallokering av dynamiskt allokerat minne.

```
update_<modulnamn>_win(op, sys_type, sing_type,
    value,mult, diff)
int         op, sys_type, sing_type;
Complex     value;
Count_ptr   mult, diff;
```

Denna rutin anropas varje gång någon förändring har skett i någon av överföringsfunktionerna eller vid markering. Här följer en beskrivning av de olika parametrarna:

op	Anger vilken typ av operation det gäller.
sys_type	Denna parameter kan anta värdena OPEN_LOOP och CLOSED_LOOP som anger vilket av de båda systemen uppdateringen gäller.
sing_type	Anger om uppdateringen berör en pol (POLE), nollställe (ZERO), förstärkningen (AMP) eller både poler och nollställen (POLE_AND_ZERO).
value	Realdelen av polen/nollställets placering anges i "value.re" och imaginärdelen i "value.im". Observera att om polen/nollstället är kompexkonjugerat skickas bara värdet med positiv imaginärdel. I fallet att det är förstärkningen som skall ändras finns det nya värdet i "value.re" medan det gamla ligger i "value.im".
mult	Talar om hur många poler, mult.n[POLE], och hur många nollställen, mult.n[ZERO], det finns i en viss position.

diff Anger skillnaden mellan det nya och det gamla antalet singulariteter i en viss position.

Det är lämpligt att utforma funktionen som en "switch"-sats med op som argument, dvs beroende på värdet av op så väljer man ett av flera alternativ. Detta eftersom det är parametern op som bestämmer hur de övriga skall tolkas och om de över huvud taget har något värde. En närmare beskrivning av vilka värden op kan anta följer här:

BEGIN_UPDATE
Skickas innan någon annan uppdatering skickas. Detta för att programmet skall kunna jobba i "batch", dvs arbeta mot en icke synlig minnesarea för att sedan kopiera hela arean till skärmen vid uppdateringens slut, om det skulle vara många olika uppdateringar. Endast op är definierad i detta fall.I [11] beskrivs noggrannare hur batch-mekanismen fungerar.

END_UPDATE
Skickas när alla uppdateringar är klara. Kan användas för att stänga av batch och därmed visa förändringarna i fönstren. Endast op är definierad i detta fall.

SET_MARK
Skickas då en pol, ett nollställe eller konstanten har markerats. Alla parametrar utom diff är definierade.

REMOVE_MARK
Skickas då markeringen skall avlägsnas. Alla parametrar utom diff är definierade.

UPDATE_VALUE
I detta fall kan sing_type anta värdena AMP eller POLE_AND_ZERO. I det första fallet anger mult antalet poler och nollställen medan diff innehåller skillnaden mot tidigare. I det senare fallet är mult och diff odefinierade.

Om det går att mata in eller ändra överföringsfunktionerna i modulen skall man komma ihåg att **inte** anropa modulens presentationfunktioner direkt utan låta kärnan sköta om detta. Uppdateringen av modulen kommer då att ske på samma sätt som uppdateringen av de andra modulerna. Fördelen med att separera inmatning och presentation på detta sätt, är att en modul till en början kan bestå av endast presentation i fönstret och att inmatning kan läggas till efter hand. Detta underlättar felsökning, eftersom en modul kan fås att fungera hjälpligt med mycket små medel.

9.3 Uppdatering av överföringsfunktionerna

Kärnan innehåller bl a funktioner för att uppdatera överföringsfunktioner och omvandla från ett öppet till ett slutet system och tvärt om. När någon ändring inträffar, genom att en modul anropar någon av de funktioner som beskrivs nedan, uppdateras både det öppna och det slutna systemet. Sedan skickas uppdateringsmeddelanden till samtliga aktiva moduler, även till den som begärde uppdateringen. Anledningen till detta är att kärnan slipper hålla reda på vem som begärde uppdateringen, det viktiga är att alla moduler uppdateras.

De funktioner som kan användas av en egen presentationsmodul för att förändra någon av överföringsfunktionerna beskrivs nedan.

```
int reset_transfer_function(rh)
Repwin_handler     rh;
```

Skickar uppdateringsmeddelande för att återställa överföringsfunktionen till starttillståndet. Uppdateringarna skickas till det fönsterpar som anges av parametern rh.

In:	rh	
Ut:	-	
Returnerar:	NO_ERR	

```
int set_K(sys_type, value)
int        sys_type;
double     value;
```

Sätter värdet på K för överföringsfunktionen i angivet system.

In:	sys_type	OPEN_LOOP eller CLOSED_LOOP
	value	nytt värde på K.
Ut:	-	
Returnerar:	NO_ERR	
	ILLEGAL_DATA	otillåtet värde på sys_type
	TOO_MANY_ZEROS	grad(B) > grad(A)
	MEMORY_FULL	för många poler/nollställen
	ZERO_DEVISOR	B = 0

```
int new_singularity(sys_type, sing_type,
    value, number)
int        sys_type, singtype, number;
Complex    value;
```

Lägger till en eller flera nya singulariteter till angiven överföringsfunktion.

In:	sys_type	OPEN_LOOP eller CLOSED_LOOP
	sing_type	POLE eller ZERO
	value	anger det komplexa värdet för polen eller nollstället.
	number	antalet singulariteter som skall läggas till.
Ut:	-	
Returnerar:	NO_ERR	
	ILLEGAL_DATA	felaktiga parametervärden
	TOO_MANY_ZEROS	grad(B) > grad(A)
	MEMORY_FULL	för många poler/nollställen
	ZERO_DEVISOR	B = 0

```
int delete_singularity(sys_type, sing_type,
    value,number)
int        sys_type, sing_type, number;
Complex    value;
```

Tar bort ett antal poler eller nollställen ur en överföringsfunktion. Om det angivna antalet skulle vara större än det befintliga returneras en felstatus.

In:	sys_type	OPEN_LOOP eller CLOSED_LOOP
	sing_type	POLE eller ZERO
	value	anger det komplexa värdet för polen eller nollstället.
	number	antalet poler/nollställen som skall tas bort.
Ut:	-	
Returnerar:	NO_ERR	
	ILLEGAL_DATA	felaktiga parametervärden
	TOO_MANY_ZEROS	grad(B) > grad(A)
	MEMORY_FULL	för många singulariteter
	ZERO_DEVISOR	B = 0

```
int move_singularity(sys_type, sing_type,
    old_value, new_value, number)
int       sys_type, sing_type, number;
Complex   old_value, new_value;
```

Flyttar ett antal poler eller nollställen i angiven överföringsfunktion. Om det angivna antalet är större än det befintliga returneras felstatus.

In:	sys_type	OPEN_LOOP eller CLOSED_LOOP
	sing_type	POLE eller ZERO
	old_value	det gamla komplexa värdet .
	new_value	det nya komplexa värdet.
	number	antalet singulariteter som skall flyttas.
Ut:	-	
Returnerar:	NO_ERR	
	ILLEGAL_DATA	felaktiga parametervärden
	TOO_MANY_ZEROS	grad(B) > grad(A)
	MEMORY_FULL	för många poler/nollställen
	ZERO_DEVISOR	B = 0

```
int remove_mark()
```

Om någon pol eller något nollställe är markerat tas denna markering bort. Observera att eventuell markering alltid automatiskt tas bort vid förändringar i överföringsfunktionerna.

In:	-
Ut:	-
Returnerar:	NO_ERR

```
int mark_singularity(sys_type, sing_type, value)
int       sys_type, sing_type;
Complex   value;
```

Markerar en pol eller ett nollställe. Om det inte skulle finnas någon singularitet med det angivna värdet returneras felstatus.

In:	sys_type	OPEN_LOOP eller CLOSED_LOOP
	sing_type	POLE eller ZERO
	value	Komplexvärdet för den singularitet som skall markeras
Ut:	-	
Returnerar:	NO_ERR	
	ILLEGAL_DATA	felaktiga parametervärden

```
int get_singularity(sys_type, sing_type, number,
    coordinates, n)
int         sys_type, sing_type, number;
Complex     *coordinates;
int         *n;
```

Denna funktion kan användas om man skulle ha behov av att explicit begära att få någon del av en överföringsfunktion.

In:	sys_type	OPEN_LOOP eller CLOSED_LOOP
	sing_type	POLE, ZERO eller AMP
	number	Ordningsnummer från ett och uppåt
Ut:	coordinates	Om sys_type = AMP returneras K i realdelen
	n	Singularitetens grad
Returnerar:	NO_ERR	
	ILLEGAL_DATA	number är för stort

```
int update_new_repwin(rh)
Repwin_handler     rh;
```

Skickar alla uppdateringar som behövs för att en nyaktiverad modul skall få all information om de aktuella överföringsfunktionerna. Denna rutin utgår ifrån att mottagaren har ett system som kan symboliseras av ett avbrott dvs båda överföringsfunktionerna är noll. Denna funktion kan användas tillsammans med reset_transfer_function() om man vill att en modul skall erhålla alla data på nytt. Observera att vid aktivering av en modul sker all uppdatering automatiskt och modulen skall då alltså inte begära att bli uppdaterad.

In:	rh	Hanteraren för det aktuella fönstret
Ut:	-	
Returnerar:	NO_ERR	

9.4 Övriga hjälprutiner

```
int dequal(d1, d1)
double     d1, d2;
```

Denna rutin använd för att jämföra två flyttal i dubbel precision med en given noggrannhet.

In:	d1 och d1
Ut:	-
Returnerar:	TRUE eller FALSE

```
int cequal(c1, c2)
Complex    *c1, *c2;
```

Jämför två komplexa tal enligt en given noggrannhet. Observera att det är pekare till de komplexa talen som skall skickas som argument.

In:	c1	pekare till ett komplext tal
	c1	pekare till ett komplext tal
Ut:	-	
Returnerar:	TRUE eller FALSE	

```
int get_polynomial(sys_type, sing_type, p)
int        sys_type, sing_type;
Polynomial     *p;
```

Hela systemet arbetar med en singularitetsbaserad representation. Det kan dock hända att man vill ha polerna eller nollställena på polynomform i stället.

In:	sys_type	OPEN_LOOP eller CLOSED_LOOP
	sing_type	POLE, ZERO eller AMP
Ut:	p	pekare till strukturen Polynomial eller, om sing_type är satt till AMP, en pekare till double.
Returnerar:	NO_ERR	
	ILLEGAL_DATA	Felaktiga parametervärden.

```
int max_repwin_height(rh)
```

Returnerar den maximala höjd ett fönsterpar kan anta.

In:	rh
Ut:	-
Returnerar:	Maximala fönsterhöjden.

9.5 Listhanteringspaketet

Vi har skrivit ett paket för hantering av dubbellänkade listor. För en beskrivning av hur listhanteringspaketet är uppbyggt hänvisas till []. Dock har vi gjort vissa nödvändiga modifieringar för att man ska kunna ha element av olika typ i listan. De nödvändiga definitionerna finns i filen "list.h". Denna fil behöver man inte tänka på om man gör "include" på filen "visidyn.h". Detta eftersom "visidyn.h" i sin tur gör "include" på "list.h".

Då man skall använda list-paketet deklarerar man först en pekare på det object man vill lägga in i listan. I alla objekt måste plats reserveras för länkarna. Exemplet nedan belyser hur det kan se ut.

```
typedef struct {
    Links l;  /*  måste finnas först i alla
                  elementdefinitioner */
/* Här definierar användaren sin datastruktur */
} Link_data;
typedef Link_data  *Link_Ptr;


Link_Ptr   pekare;
Head       min_lista;
```

Det är sedan pekaren och listans namn som används som argument till alla operationer. För en närmare beskrivning av de olika operationer som finns på listan hänvisar vi till filen "list.h".

9.6 Meddelanden

Det finns tillfällen då man vill göra användaren uppmärksam på något eller vill ha en bekräftelse på ett kommando. För detta ändamål finns det två rutiner som anropas på följande sätt:

```
int  svar;

/* returnerar OK */
svar = visi_message(text_1, ..., text_n, 0);

/* returnerar OK eller CANCEL */
svar = visi_confirm(text_1, ..., text_n, 0);
```

Observera att argumentlistorna måste avslutas med en nolla.

Dessa funktioner tar ett godtyckligt antal strängar som argument och skriver ut strängarna i ett separat fönster. Används det första anropet finns det bara ett sätt för användaren att kvittera meddelandet och fortsätta programkörningen. Om däremot det andra anropet används kan användaren välja mellan att godkänna eller avbryta operationen.

För att få tillgång till de nödvändiga deklarationerna måste filen "message.h" inkluderas i programmet.

9.7 Installering av ny modul

För att kärnan skall få reda på att en ny modul finns tillgänglig, så att den kan bli valbar i huvudpanelen, måste man lägga till ett nytt anrop av funktionen init_repwin() i filen "visidyn_conf.c". Anropet skall se ut på följande sätt:

```
init_repwin(
     /* Namnet som kommer upp i huvudpanelen */
     "name",
     /* Ett unikt positivt heltal */
     number,
     /* Typ av fönster, t ex CANVAS */
     win_type,
     /* Fönstrets minsta tillåtna höjd i pixel */
     min_height,
     /* Fönstrets största tillåtna höjd i pixel */
     max_height,
     /* Procedur som anropas då fönstret skapas */
     create_win,
     /* Anropas då fönstret tas bort */
     delete_win,
     /* Anropas vid uppdateringar */
     update_win
     );
```

För att på ett kunna använda kommandot "make" måste filen "makefile" uppdateras så att den nya modulen finns med. Hur "make" fungerar står beskrivet i de flesta böcker om UNIX, t ex [4], eller i [10]. Efter att man har kompilerat och länkat programmet med "make" känner kärnan till den nya modulen och den kan väljas på samma sätt som de andra i programmets huvudpanel.

10 Sammanfattning

Syftet med arbetet har varit att konstruera ett verktyg för interaktiv design av överföringsfunktionen i enkelt återkopplade system. Tyngdpunkten har lagts på de grafiska aspekterna. Vi valde därför att begränsa oss till lineära system med endast en insignal och en utsignal. Dessutom fick inga tidsförskjutningar förekomma. Programmet skrevs i C och körs under Suntools på en arbetsstation av typ Sun 3/50.

Programmet består av en huvudpanel som visas då programmet startas. I denna panel kan man bl a välja vilka presentationsformer som skall visas. En överföringsfunktion kan för närvarande presenteras på tre olika sätt:

- Singularitetsdiagram
- Bodediagram
- Numeriskt i form av poler och nollställen

Det är dessutom enkelt att lägga till nya moduler för t ex stegsvar eller Nyquistdiagram.

Det fattas ännu en del för att programmet skall vara användbart vid konstruktion av regulatorer. En viktig bit som saknas är att kunna "låsa" singulariteter. Alltså att tala om vilka singulariteter som hör till processen och inte kan ändras. Vi har försökt att strukturera och modularisera vår kod på ett sådant sätt att det skall vara enkelt att lägga till nya funktioner. Programmet skall alltså kunna utgöra en bas för fortsatt utveckling.

Att vi skulle arbeta med ett fönsterhanteringssystem var klart från början. Dessa innehåller de grafiska möjligheter och ger den flexibilitet i presentationen som krävdes för arbetet. Vi valde att arbeta med SunView eftersom detta var det enda system som vi hade tillgång till då arbetet påbörjades. Det har visat sig att det finns ett bra stöd för hantering av paneler, texter, grafik mm. Tyvärr visar de olika modulerna upp inbördes olikheter t ex vid textinmatning. För övrigt kan sägas att det är mycket enkelt att göra mindre applikationer under SunView. Mera komplicerade saker, t ex ändring av storleken på underfönster, kan vara ganska besvärligt.

Det system som skulle vara det främsta alternativet är X Windows. Detta på grund av att detta system ser ut att bli en standard. Vad som talar emot X Windows är att det inte, så vitt vi vet, finns något stöd i form av färdiga rutiner för paneler etc. Hade detta funnits skulle vårt val i dag fallit på X Windows.

Referenser

[1] Becker, Hans, Fönster styrs med laserskrivarspråk, industriell Datateknik,
Nr 6/april 1987

[2] Ekman, Torgil, Karlsson, Jan, Programmering och datastrukturer med Pascal, Studentlitteratur, Lund, 1983

[3] Glad, Torkel, Ljung, Lennart, Reglerteknik-Grundläggande teori, Studentlitteratur, Lund, 1981

[4] Kernighan, Brian W., Pike, Rob, The UNIX Programming Environment, Prentice-Hall, 1984

[5] Kernighan, Brian W., Ritchie, Dennis M., The C Programming Language, Prentice-Hall, 1978

[6] Stern, Hal L., Comparison of Windowing Systems, BYTE, November 1987

[7] Sun Microsystems Inc, Commands Reference Manual, 1986

[8] Sun Microsystems Inc, FORTRAN Programmers Guide, 1986

[9] Sun Microsystems Inc, Pixrect Reference Manual, 1986

[10] Sun Microsystems Inc, Programming Utilities for the Sun Workstation, 1986

[11] Sun Microsystems Inc, SunView Programmer´s Guide, 1986

[12] Sun Microsystems Inc, UNIX Interface Reference Manual, 1986

[13] Sun Microsystems Inc, Windows and Window Based Tools: Beginner´s Guide, 1986

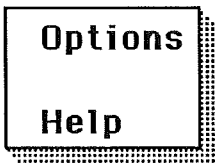
[14] Wenban, Alan S., Poles Zeros and Timeresponse: A Computer-Aided Instructional Program for Relating a System´s Transfer Function to its Time-domain Response, Renselaer Polytechnic Institute, Troy New York, May 1987

[15] Åström, Karl Johan, Reglerteori, Almqvist & Wiksell Förlag AB, Stockholm, 1976

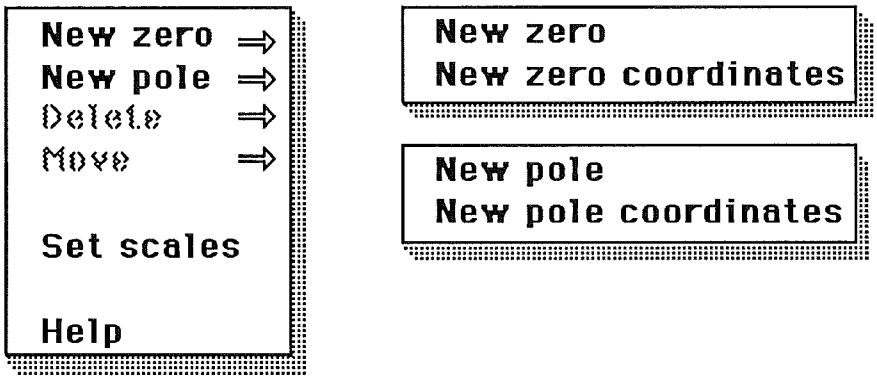
[16] Åström, Karl Johan, Wittenmark, Björn, Computer-Controlled Systems - Theory and Design, Prentice-Hall, 1984

Appendix A - Menysammanställning

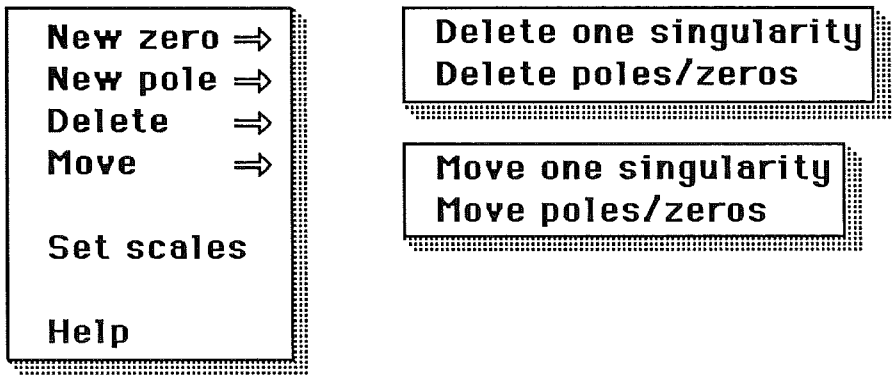
I detta appendix visar vi hur samtliga menyer som finns i VISIDYN ser ut.



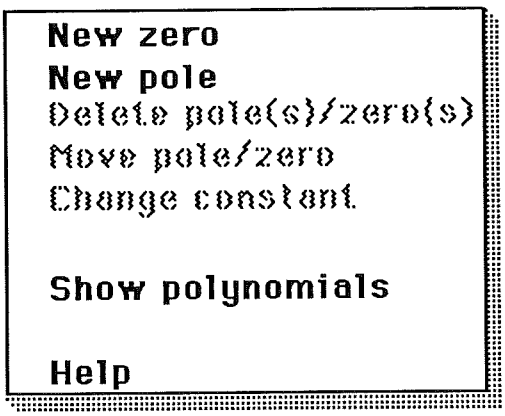
Huvudpanel

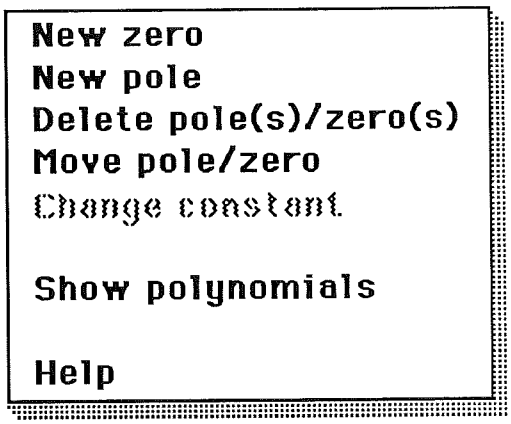


Singularitetsdiagram då det inte finns någon markerad singularitet

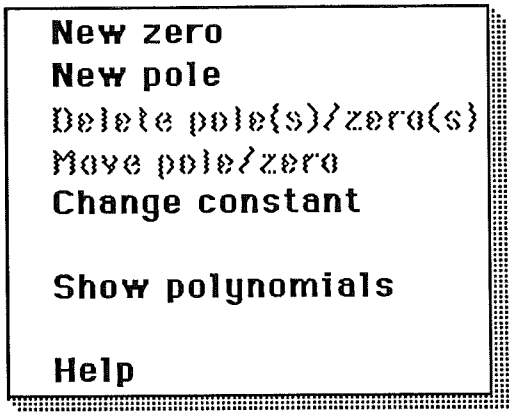


Singularitetsdiagram med markerad singularitet

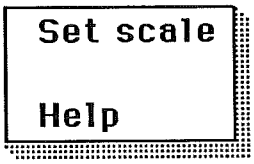




Överföringsfunktionsfönstret utan markering respektive med markerad singularitet



Överföringsfunktionsfönstret med förstärkningsfaktorn markerad



Bodediagram

Appendix B - Exempel

Det är svårt att i text åskådliggöra ett interaktivt program. En demonstration ger mycket mer. I detta kapitel ges i alla fall en kortfattad beskrivning av ett enkelt exempel på användning.

Exempel 1 - Undersökning av ett enkelt system

Givet är ett system med följande öppna överföringsfunktion:

$$G_o(s) = \frac{1}{s\,(s+1)\,(s+2)}$$

Vi är intresserade av att se det slutna systemets egenskaper. Definiera först överföringsfunktionen på följande sätt:

1) Markera i huvudpanelen alternativen "Singularity diagram", "Transfer function" och "Bode diagram".

2) Placera ut singulariteterna genom att välja lämpligt alternativ i singularitetsdiagrammets fönster eller i fönstret för numerisk visning av överföringsfunktionen. I singularitetsdiagrammet flyttas musen tills önskade koordinater står ovanför ramen och sedan trycks vänsterknappen ned. I det numeriska fönstret fås en dialogruta där koordinaterna skrivs in.

Vi ska nu bestämma egenskaperna för systemet genom mätning i Bodediagrammet. Mätningen sker genom att musens mittenknapp trycks ned i antingen fas- eller beloppkurvan. När sedan musen flyttas i sidled uppdateras de avlästa värdena ovanför beloppkurvan. Genom att gå in och mäta i Bodediagrammet för det öppna systemet kan vi bestämma amplitud- och fasmarginal:

Amplitudmarginalen

Genom att gå in och läsa av beloppkurvans värde i den punkt där fasvridningen är -180 grader kan man bestämma amplitudmarginalen. I detta fall är värdet i denna punkt -0.78 och vi räknar ut amplitudmarginalen på följande sätt

$$A_m = 10^{0.78} \approx 6$$

Detta inträffar vid vinkelfrekvensen 1.423 vilket skall jämföras med det teoretiska värdet som är $\sqrt{2}$.

Fasmarginalen

Denna kan bestämmas genom att mäta i faskurvan i den punkt där amplitudmarginalen är noll. I detta fall är faskurvans värde -126 grader och då blir fasmarginalen 180-126=54 grader. Även detta värde stämmer mycket väl överens med det korrekta.

Genom att ändra förstärkningen kan ett slutet system med önskade egenskaper erhållas. Förstärkningen ändras genom att värdet som står under rubriken "Const" i fönstret för den numeriska visningen av överföringsfunktionen

markeras med vänster musknapp varefter alternativet "Change constant" i fönstrets meny väljs. I dialogrutan som då visas på skärmen skrivs den nya förstärkningen in. Sätter man förstärkningen lika med amplitudmarginalen skall man t ex få ett slutet system som ligger på stabilitetsgränsen. Att detta också är fallet visar skärmbilden i Appendix C.

Exempel 2 - Reglering av tank

Detta exempel skall belysa hur VISIDYN kan användas för att bedöma de prestanda som kan uppnås med enkla regulatorer. Exemplet anknyter till laborationer i Reglerteknik AK vid LTH. Först studeras PI-reglering av vattennivån i en tank. Vattentillflödet skall regleras så att en konstant vattennivå erhålles oberoende av utflödet ur tanken. Överföringsfunktionen för en tank kan, med lämpligt val av enheter, beskrivas som

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

Processen har således en tidskonstant på 1 sekund. I verkligheten svarar detta mot ca 50 s. Vid reglering vill vi att det slutna systemet skall vara betydligt snabbare, låt säga en tidskonstant på 0.2 s (detta svarar mot ca 10 s i verkligheten). En PI-regulator har överföringsfunktionen

$$G_R(s) = k\frac{s + a}{s} = k\left[1 + \frac{a}{s}\right]$$

där $a = 1/T_i$ och T_i är integraltiden.

Parametern a kallas regulatorns nollställe och k dess förstärkning. Det öppna systemet har således överföringsfunktionen

$$G(s) = k\frac{s + a}{s(s + 1)}$$

Vi är intresserade av att ta reda på hur systemet påverkas av parametrarna a och k. Speciellt vill vi undersöka om a och k kan väljas så att det slutna systemet får poler i $-2\pm2i$. Detta val ger ett slutet system med väl dämpat svar.

Börja med att sätta ut de båda polerna i det öppna systemet. Den ena i origo och den andra i -1. Parametern a anger nollställets placering på den reella axeln. Genom att placera ut nollstället och sedan flytta det längs reella axeln fås en god uppfattning om nollställets inverkan på det slutna systemet. Då nollstället befinner sig **mellan** de båda polerna i det öppna systemet är samtliga singulariteter i det slutna systemet reella. För att få komplex-konjugerade poler i det slutna systemet måste alltså nollstället i det öppna systemet befinna sig till **vänster** om **båda** polerna. Observera också att endast polernas avstånd till reella axeln kan påverkas av a. Det går inte att ändra deras realdel genom att flytta nollstället.

Då nollstället nu är utplacerat till vänster om de båda polerna skall vi se vad som händer då förstärkningen k ändras. Genom att långsamt öka k kan man se hur polerna i det slutna systemet rör sig i en halvcirkel runt nollstället. Genom att ändra både a och k kan polerna i det slutna systemet alltså placeras fritt.

För att få polerna i det slutna systemet att ligga i -2±2i väljer man a≈2.7 och k≈3. Eftersom nollstället har en realdel som ligger mycket nära de dominerande polernas realdel får man överväga att minska börvärdets inflytande på stegsvaret.

Det går alltså utmärkt att reglera vattennivån i en tank med hjälp av en PI-regulator. Vi utökar nu exemplet och låter utflödet från den ovan beskrivna tanken rinna ner i en annan likadan tank. Det är nu nivån i den andra tanken som skall regleras. Denna process har överföringsfunktionen

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

Om vi vill reglera detta system med en PI-regulator blir systemets överföringsfunktion

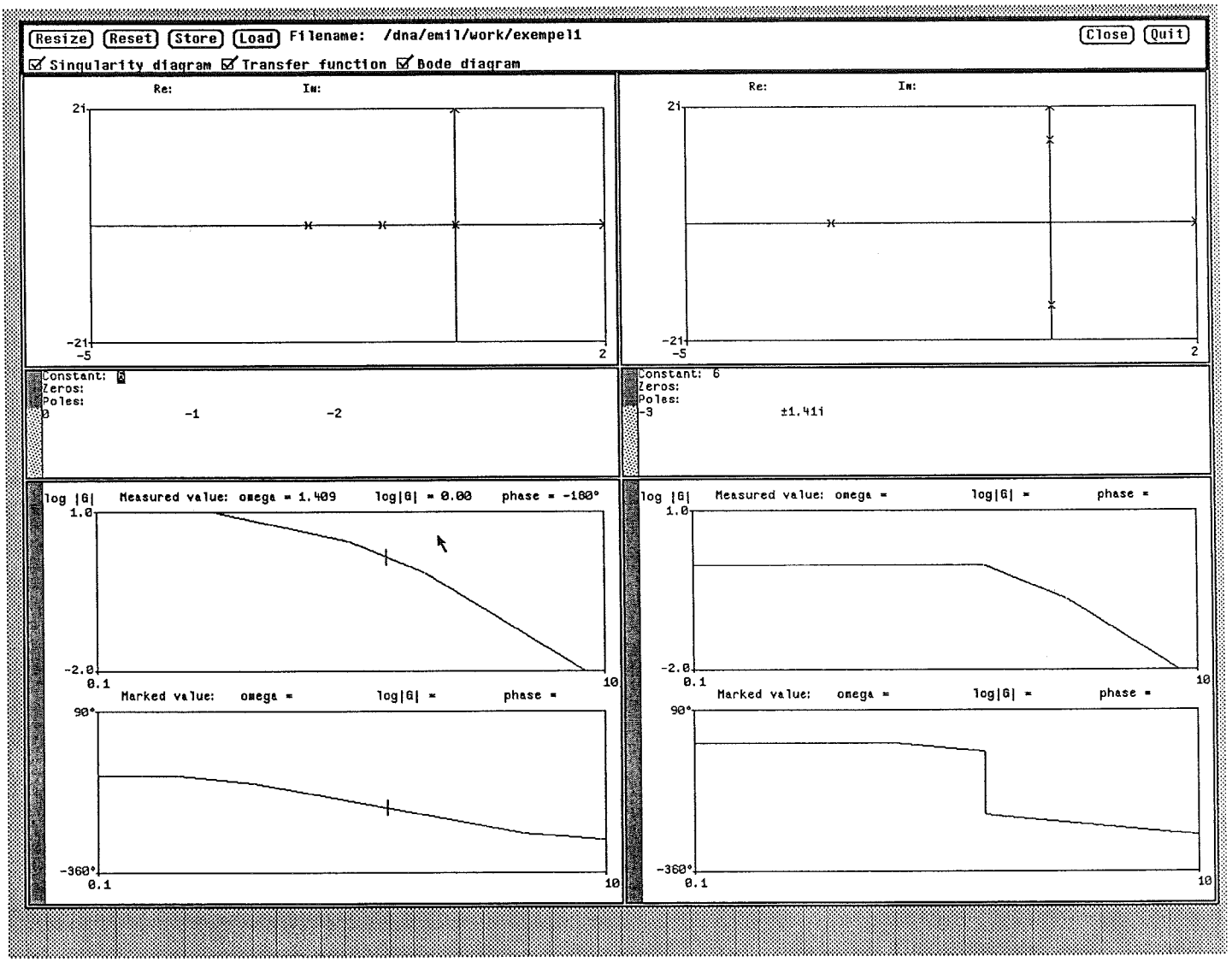
$$G(s) = k\frac{s+a}{s(s+1)^2}$$

Vi skall nu använda vårt system för att experimentellt undersöka vad som kan göras. Placera först ut systemets poler. Försök sedan att placera två av det slutna systemets poler i -2±2i genom att ändra på a och k. Detta visar sig vara omöjligt. I själva verket kan man inte få polernas realdel mindre än -1. Detta illustrerar svårigheterna med att reglera två tankar med en PI-regulator. Det går visserligen att finna en regulatorinställning som ger ett väl dämpat system men det slutna systemet blir långsamt därför att polernas realdel är större än -1. Derivatakompensering är nödvändig för att få ett snabbare system.

Appendix C - Skärmbild

Här nedan finns ett exempel på hur det kan se ut då man arbetar med programmet. Överst syns huvudpanelen med de övergripande funktionerna. Under detta kommer ett antal fönsterpar innehållade olika representationer av samma överföringsfunktion. De fönsterpar som visas är ovanifrån:

- Singularitetsdiagram
- Förstärkningen och poler/nollställen på numerisk form
- Bodediagram



Den överföringsfunktion som visas är hämtad ur exempel 1. Där undersöks hur stor förstärkning systemet tål utan att bli instabilt.

Index