SIMNON
Ett interaktivt simuleringsprogram för olinjära system

Hilding Elmqvist

RE-113

SIMNON

Ett interaktivt simuleringsprogram för olinjära system

Examensarbete utfört av

Hilding Elmqvist

Handledare: Johan Wieslander

Institutionen för Reglerteknik
Lunds Tekniska Högskola

ABSTRACT

This report includes:

A definition of a language suitable for the description of nonlinear systems.

Development of FORTRAN-routines that performs a compilation of the language to a pseudo-code and a routine to interprete and execute this code.

The welding of the compiler and the execution-routine into a complete interactive simulation-package.

SAMMANFATTNING

Detta examensarbete omfattar:

Definition av ett språk lämpat att beskriva olinjära system.

Utarbetande av FORTRAN-subrutiner (kompilator) som utför översättning av språkelementen till en exekverbar pseudokod samt en rutin för exekvering av denna kod.

Infogning av kompilatorn och exekveringsrutinen i ett interaktivt simuleringspaket.

INNEHÅLLSFÖRTECKNING

		sid
I.	Introduktion	1
II.	Definition av simuleringsspråk	3
III.	Kompilator och exekveringsrutin	5
IV.	Simuleringspaket	16
V.	Exempel på användning av SIMNON	22

Appendix 1. Syntax för simuleringsspråk

Appendix 2. Felmeddelanden vid kompilering

Appendix 3. Syntax för kommandon

I. INTRODUKTION

Målet för detta examensarbete har varit att skapa ett simuleringspaket, med vars hjälp man kan simulera olinjära system på dator.

Den aktuella datorn är PDP-15 på institutionen för Reglerteknik, som bl.a. är utrustad med skrivmaskin (teletype), bildskärm (display) och skivminne (i fortsättningen kallat disken).

För att så enkelt som möjligt kunna beskriva olinjära system har valts att göra detta med hjälp av första ordningens differentialekvationer

$$\frac{dx}{dt} = f(x,u,t)$$

x tillståndsvektorn
u insignalvektorn
t tiden.

Detta till skillnad från vissa simuleringspaket, som för beskrivningen utgår från ett blockschema med integratorer.

Inläsning av systembeskrivningen skall kunna göras antingen från teletypen eller disken.

För att integrera system av första ordningens differentialekvationer finns tillgänglig en FORTRAN-subrutin DHAMDI[1)], som även klarar diskontinuiteter. Under integreringen behöver DHAMDI veta värdet av $\frac{dx}{dt}$ för givna x, u och t och anropar därför subrutinen DERIV. I detta fall skall beräkningen av $\frac{dx}{dt}$ ske enligt den symboliska beskrivningen av systemet. Under en simulering behöver beräkningarna utföras många gånger (100 - 1000), varför det skulle vara alltför tidskrävande att tolka den symboliska beskrivningen varje gång. I stället översätts ekvationerna (kompileras) till en enkel pseudokod lämplig för exekvering.

1) DHAMDI redovisas i:
Göran Fick, DHAMDI a FORTRAN subroutine to integrate a set of first order, ordinary differential equations containing discontinuities
FOA 2 rapport C 2504 - E5 November 1971

I vissa punkter anropar DHAMDI subrutinen OUTP, som är en utmatningsrutin. Ovannämnda kod genomlöps även då, varefter plottning av valda variabler kan ske på displayet.

I nästa kapitel redovisas hur systembeskrivningen skall göras. I tredje kapitlet visas hur kompileringen utföres, samt de subrutiner som utför detta och utför exekveringen. Fjärde kapitlet ägnas åt simuleringspaketets uppbyggnad, vilka kommando som finns och vad dessa medför. Kapitel fem innehåller exempel.

II. DEFINITION AV SIMULERINGSSPRÅK

1. Inledning

Det primära i simuleringsspråket är ekvationer av typen

$$\frac{dx_1}{dt} = f_1(x_1,x_2,\ldots,x_n,t)$$

Denna form av indicering av tillståndsvariabler är emellertid omöjlig på teletypen, varför x_1 skrives X1. Som beteckning för derivatan har valts DX1.

I funktionen f_1 kan ingå:

1. uttryck som kan skrivas med hjälp av + - * / ↑ ( )
2. standardfunktioner av typ SIN och COS
3. egna funktioner (kallade FCN) som anges i punktform, varvid interpolering sker vid exekvering
4. IF-satser av typ IF boolskt uttryck THEN uttryck ELSE uttryck (samma som i ALGOL).

För att vid simulering kunna se inverkan av ändringar av vissa parametrar kan man i stället för numeriska konstanter införa identifierare för parametrar. Dessa tilldelas värden med satser av form PAR:1. Identifierare för egna variabler kan likaså införas.

För insignaler och utsignaler har reserverats beteckningar av typ U1 resp. Y1 och för tiden T.

2. Syntaktisk definition

I Appendix 1 redovisas en syntaktisk definition av simuleringsspråket. Där använda beteckningar har följande betydelse:

:=	utgöres av
/	eller
$\{arg\}_i^j$	arg upprepas minst i och högst j antal gånger, då i och j är utelämnade skall detta tolkas som i=1 och j=1
---	dock ej (undantaget)

Nedan följer vissa kommentarer till syntaxen.

1. <one argument function>/<two argument function> har exakt samma betydelse som FORTRAN:s funktioner.
2. Jämförelser med ALGOL:
   a. <number>
      $_{10}$ är ersatt av E,
      <unsigned number> kan ej utgöras av enbart <exponent part>,
      <unsigned number> kan utgöras av <unsigned integer>.
      (dvs. <unsigned integer> följt av punkt som i FORTRAN).
   b. <arithmetic expression>
      <multiplying operator> kan ej utgöras av ÷.
   c. <boolean expression>
      $\leq \neq \geq \supset \equiv$ är ej tillåtna,
      logiskt eller, och, icke skrives OR, AND, NOT,
      <boolean primary> kan ej utgöras av <logical value>/
      <function designator> men kan utgöras av <parameter>.
3. <own function definition> göres i tabellform med varje par av <argument value><function value> på en rad.
4. <comment> kan utgöras av enbart <line terminator>.
5. I <system description> kan ingå <X0 definition>/<parameter definition>/<own function definition>/<assignment statement>/<comment>. Avslutning göres med *END.
6. Blanktecken behandlas ej av den syntaktiska definitionen, men måste infogas bl.a. mellan <boolean word> och <identifier>. Blanktecken får infogas var som helst utom i <name>/<reserved word>/<unsigned number>/FIN/END.

III. KOMPILATOR OCH EXEKVERINGSRUTIN

1. Inledning

Kompilatorn skall utföra översättning av <system description>. Därvid är <X0 definition>/<parameter definition>/<own function definition> väsentligen en fråga om att lagra in numeriska värden på bestämda platser. Ett svårare problem är att alstra exekverbar kod utgående från ett <assignment statement>.

Den exekverbara koden utgöres av REVERSE POLISH NOTATION (i fortsättningen förkortat RPN). Vid exekvering av RPN används en stack (vektor) S för att lagra mellanresultat. Hur denna stack används förklaras enklast med ett exempel.

Exempel 1.
Vid exekvering av RPN för
A=B+2*(C+1)-3
utföres

S_1:=adress(A)
S_2:=B
S_3:=2
S_4:=C
S_5:=1
S_4:=S_4+S_5 =C+1
S_3:=S_3*S_4 =2*(C+1)
S_2:=S_2+S_3 =B+2*(C+1)
S_3:=3
S_2:=S_2-S_3 =B+2*(C+1)-3
*S_1:=S_2

Den sista operationen på stacken betyder att värdet i S_2 överföres till den adress som finns i S_1.

Den aktuella kompilatorn arbetar i två steg vid behandling av <assignment statement>. Först kodas den inkomna teckensträngen och därefter alstras RPN.

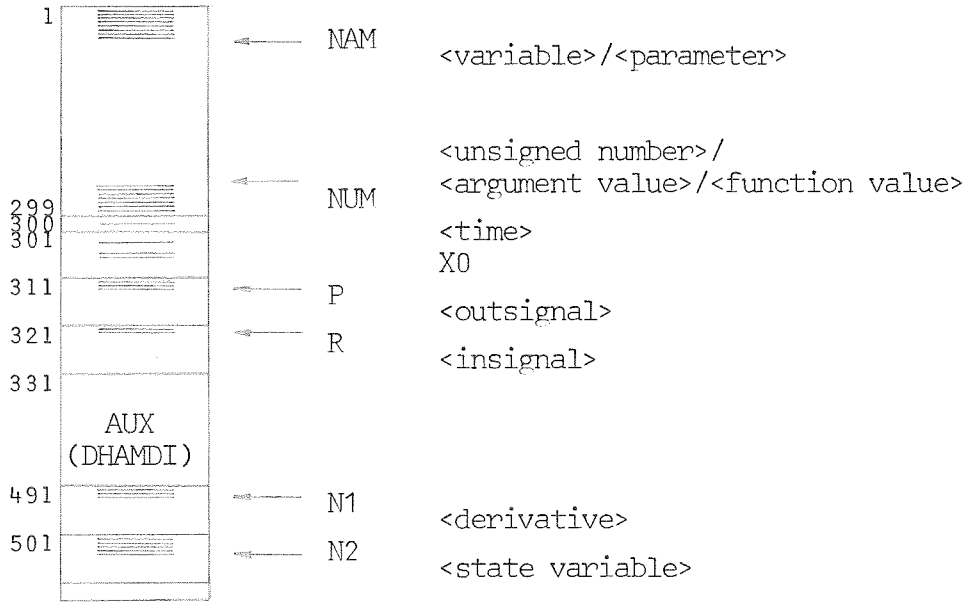
2. Kodning

Vid kodningen ersätts <identifier> med den adress där motsvarande värde skall lagras. Om <identifier> utgörs av <name> undersöks därvid om samma <name> använts tidigare. På motsvarande sätt ersätts <unsigned number> med den adress där talet lagras. Likaså kodas operatorer och återstående <reserved word>.

Koden lagras i en heltalsvektor COD där varje element består av en operationsdel och en adressdel och beräknas som

COD(K)=OP*1000+ADR.

Adressen (ADR) hänför sig till en reell vektor VALUE(512) som disponeras enligt fig. 1.



Figur 1.

Ur fig. 1 ses att antalet <state variable>/<derivative>/<insignal> /<outsignal> har maximerats till 10.

Lagring av <name> sker i en vektor NAM(134). Dessutom används en vektor KIND(134) för att veta om <name> är <variable>/<parameter> samt för att indikera om <parameter> är odefinierad. NAM och KIND kan tänkas ligga parallellt med början av VALUE.

Operationsdelens (OP) betydelse efter kodning framgår av tabell 1.

Tabell 1: Operationsdelens (OP) betydelse efter kodning.

OP	
0	<identifier>/<unsigned number>
1	OR
2	AND
3	NOT
4	<
5	>
6	= (relation)
7	+
8	-
9	*
10	/
11	- (unitärt)
12	↑
13	IF
14	THEN
15	ELSE
16	= (tilldelning)
17	(
18	((efter funktioner)
19	)
20	,
21	" eller <line terminator>
22	<one argument function>/<two argument function>
23	<own function>

Ur tabell 1 ses att unitärt - (Ex. A=-B) har en särskild kod (unitärt + elimineras). Vid kodningen skiljs också på vänsterparanteser. Sådan parantes som följer en funktion har särskild kod. Då OP är lika med 22 anger ADR vilken av de 12 funktionerna som är aktuell. Likaså då OP är lika med 23 anger ADR aktuell <own function> (antalet är maximerat till 10).

Nedan ges exempel på den kod som bildas vid kodning av <assignment statement>.

Exempel 2.
Om man börjar en <system description> med

A=B+2*(C+1)-3

blir innehållet i COD

1	1	A	6	9000	*	11	19000	)
2	16000	=	7	17000	(	12	8000	-
3	2	B	8	3	C	13	297	3
4	7000	+	9	7000	+	14	21000	"
5	299	2	10	298	1			

(B och C indikeras som odefinierad <parameter>)

Fortsättes sedan med

DX1=A+(IF X1>2 AND BOOL THEN SIN(T) ELSE FCN2(5+SQRT(X2)))

erhålles koden

1	491	DX1	11	4	BOOL	21	7000	+
2	16000	=	12	14000	THEN	22	22004	SQRT
3	1	A	13	22001	SIN	23	18000	(
4	7000	+	14	18000	(	24	502	X2
5	17000	(	15	300	T	25	19000	)
6	13000	IF	16	19000	)	26	19000	)
7	501	X1	17	15000	ELSE	27	19000	)
8	5000	>	18	23002	FCN2	28	21000	"
9	299	2	19	18000	(			
10	2000	AND	20	296	5			

(BOOL, DX2 och FCN2 indikeras som odefinierade)

3. Alstring av RPN

Utgående från ett kodat <assignment statement> skall RPN fyllas på med den exekverbara kod som erfodras.

Alstring av RPN kan göras på olika sätt men i detta fall används en metodik som baseras på att operatorer kan lagras undan i en

stack T. Vid behandling av nästa operator jämföres denna med översta operatorn i T. Beroende av denna jämförelse utför vissa rutiner olika förändringar av RPN och T. För att behandla funktioner har införts en särskild stack F, som också berörs av nämnda rutiner.

Jämförelsen mellan operatorerna sker med hjälp av en matris PRI(13,14) vars element är heltal 1 - 9, som anger vilken rutin som skall användas.

En mer utförlig beskrivning av metodiken återfinns i KOMPILATORTEKNIK[1]. I detta examensarbete behandlas emellertid funktioner på ett annorlunda sätt och IF-satser på ett mer anpassat sätt.

Då OP är lika med 0, svarande mot <identifier>/<unsigned number>, överförs COD(K) adderat med 18000 om utgörande av <left part> annars 17000 direkt till RPN.

Operationsdelens betydelse i RPN är då $1 \leq OP \leq 12$ helt motsvarande OP i COD, vilket exemplifieras nedan.

Exempel 3.
Då OP=7 i RPN kommer vid exekveringen de två översta elementen i exekveringsstacken S att adderas. Stackindex minskas sedan med ett och resultatet lagras i toppen.

Exempel 4.
OP=4: Om näst översta elementet i S är mindre än toppelementet erhålles resultatet 1.0 annars -1.0.

Exempel 5.
OP=1: Om näst översta elementet i S ≥ 0 eller toppelementet ≥ 0 erhålles resultatet 1.0 annars -1.0.

Allmänt gäller att om en relation är TRUE eller resultatet av en logisk operation är TRUE erhålles resultatet 1.0 annars -1.0. För att avgöra om en boolsk variabel är TRUE testas om motsvarande värde är större än eller lika med 0.

För värden på OP som är större än 12 framgår av tabell 2 vad som utföres vid exekvering.

1) Torgil Ekman Institutionen för informationsbehandling Lund 1971

Tabell 2: Exekvering av RPN(L)=OP*1000+ADR då OP ≥ 13.

ARG1 = översta stackelementet
ARG2 = näst översta stackelementet

OP	
13	om ARG1 ≥ 0 fortsätt exekveringen med nästa instruktion annars fortsätt med instruktionen i RPN(ADR) (hoppa om FALSE).
14	fortsätt exekveringen med instruktionen i RPN(ADR) (ovillkorligt hopp).
15	- (används ej).
16	överför ARG1 till VALUE(ARG2) (tilldelning) (först göres IFIX på ARG2).
17	överför VALUE(ADR) till toppen på stacken.
18	överför ADR till toppen på stacken (först göres FLOAT på ADR).
19	RETURN (exekveringen klar).
20	- (används ej).
21	- (används ej).
22	applisera den ADR:te funktionen på ARG1 och eventuellt på ARG2.
23	interpolera med ARG1 i värdetabellen för FCN nr ADR.

För att demonstrera RPN:s uppbyggnad visas RPN för de <assignment statement> vars COD visats i Exempel 2.

Exempel 6.

RPN för

```
A=B+2*(C+1)-3
DX1=A+(IF X1>2 AND BOOL THEN SIN(T) ELSE FCN2(5+SQRT(X2)))
```

blir

```
 1   18001      11   16000      21   22001
 2   17002      12   18491      22   14028
 3   17299      13   17001      23   17296
 4   17003      14   17501      24   17502
 5   17298      15   17299      25   22004
 6    7000      16    5000      26    7000
 7    9000      17   17004      27   23002
 8    7000      18    2000      28    7000
 9   17297      19   13023      29   16000
10    8000      20   17300
```

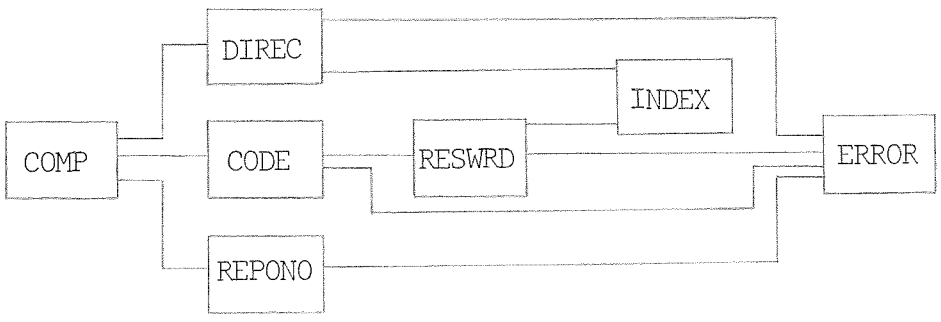
Instruktionerna 1 t.o.m. 11 svarar mot det första <assignment statement>.

4. Implementering av kompilatorn

Grundläggande för implementeringen har varit fyra biblioteks-rutiner:

RLINE	Läser en rad till en buffert
FAC	Hämtar ett tecken från bufferten
RABC	Läser en Hollerith sträng från bufferten
RFP	Läser ett tal i fritt format från bufferten

För kompileringen har skrivits sju FORTRAN-subrutiner med inbördes relationer enligt fig. 2.



Figur 2.

Nedan ges en kortfattad beskrivning av vad respektive subrutin utför.

ERROR

Sköter om eventuella felutskrifter vid kompileringen. Felutskrifterna erhålles på displayet och är i kodad form, varvid ett heltal 0 - 99 anger aktuellt felmeddelande. En förteckning över dessa återfinns i Appendix 2. Första gången ERROR anropas efter det att ny rad lästs in skrives raden på displayet. I felutskriften ingår en pil som markerar var i raden felet upptäcktes.

INDEX

Utför beräkning av värdet hos <index>.

RESWRD

När ett alfabetiskt tecken påträffas i raden anropas först RESWRD, som testar om detta och närmast följande tecken utgör <reserved word>. Om så är fallet beräknas bl.a. OP och ADR.

DIREC

Tar hand om <own function definition>, *END och *CLOSE.

<argument value> och <function value> i <own function definition> placeras i VALUE i samma område som <unsigned number>. För att vid exekveringen veta var en viss <own function> har sina värden lagrade, anges första och sista värdets placering i en heltalsvektor FCN(10), vars element beräknas som BEG*350+FIN.

När *END anges testas att alla <parameter>/<derivative>/<own function> som har använts är definierade. I så fall avslutas <system description>.

*CLOSE har införts utöver syntaxen att användas i stället för *END för att möjliggöra avslutning utan tester.

CODE

Handhar <X0 definition>/<parameter definition> och kodningen av <assignment statement>.

X0 behöver ej anges för alla <state variable>, som då förutsätts vara lika med noll.

REPONO

Fyller på RPN utgående från ett kodat <assignment statement>.

COMP

Är en samordnande subrutin.

För att veta var ett nytt <name> eller <unsigned number> skall placeras, vilka <derivative>/<insignal>/<outsignal> som definierats samt vilka <state variable> som använts har införts "pointers" med betydelse som framgår av fig. 1. Dessutom finns "pointers" för definierade <own function> och använda <own function> samt för nästa lediga element i RPN.

För varje "pointer" finns ett accepterat och ett provisoriskt värde. När en rad kompileras ändras det provisoriska värdet. Om raden var felfri accepteras det provisoriska värdet annars återställs det.

Mycket arbete har lagts ner på att få kompilatorn att upptäcka alla fel i <system description>. Det har emellertid visat sig svårt att särskilja <arithmetic expression> och <boolean expression>, varför vissa typer av uttryck, som ej är syntaktiskt riktiga, kommer att godkännas.

Exempel 7.
Nedanstående uttryck kommer att godkännas:

```
IF A+B THEN ...
IF SIN(A) THEN ...
NOT A+B
A+(B>C)
SIN((NOT A))
```

Exekveringsrutinen har emellertid gjorts så att ovanstående typer av uttryck blir meningsfulla. Jämför exempel 4 och 5.

För att få samstämmighet mellan kompilatorn och syntaxen, får tilllägg göras till två definitioner:

<primary> kan även utgöras av (<boolean expression>)

<boolean primary> kan även utgöras av <simple arithmetic expression>

Med nämnda tillägg fås emellertid en total sammanblandning av <arithmetic expression> och <boolean expression>, varför användning av sådana uttryck kan bli svårtolkad.

Sammanlagda antalet FORTRAN-satser (exkl. kommentarer) i kompilatorn är ca 650.

5. Rutin för exekvering av RPN

För att utföra exekvering av RPN har gjorts en subrutin CALCUL. Denna subrutin innehåller 20 små delrutiner som utför operationer på de översta stackelementen i enlighet med tabell 1 ($1 \leq OP \leq 12$) och tabell 2 ($13 \leq OP$).

För att demonstrera CALCUL:s struktur visas nedan en listning av vissa avsnitt.

```
1.      100   L=L+1

2.            OP=RPN(L)/1000

3.            ARG1=S(R-1)
              ARG2=S(R-2)

4.            GO TO(1,2,3,4,5,6,7,8,9,10,11,12
             :,13,14,99,16,17,18,19,99,99,22,23),OP

5.      1     RES=-1.0
              IF(ARG2.GE.0. .OR. ARG1.GE.0.) RES=1.0
              GO TO 200

6.      4     RES=-1.0
              IF(ARG2.LT.ARG1) RES=1.0
              GO TO 200

7.      7     RES=ARG2+ARG1
              GO TO 200

8.      200   R=R-1
9.      300   S(R-1)=RES
10.           GO TO 100
```

Kommentarer till angivna punkter i CALCUL:

1. Ökning av instruktionsräknaren med 1.
2. Beräkning av RPN:s operationsdel.
3. För att minska CALCUL:s kärnminnesbehov, genom att varje delrutin ej skall behöva adressera indicerade variabler, överförs översta och näst översta elementen i exekveringsstacken till ARG1 och ARG2 (stackindex R pekar på första lediga element i S).
4. Med ett väljarstyrt hopp hoppas till respektive delrutin.
5. Operation OR
6. Operation <
7. Operation +
8. Minskning av stackindex med 1 om operationen ej är unitär.
9. Överföring av resultatet till toppen på stacken.
10. Utför nästa instruktion i RPN.

Vid beräkning av <own function>:s värde testas först om ARG1 är mindre än eller större än minsta respektive största <argument value>. Om så är fallet beräknas RES från en tänkt linje genom de två yttersta punkterna. Om ARG1 ligger inom intervallet halveras detta och bestämmning göres inom vilken hälft ARG1 ligger. Halveringen upprepas tills närmast mindre och närmast större <argument value> har bestämmts, då interpolering sker.

Antalet FORTRAN-satser (exkl. kommentarer) i CALCUL är ca 100.

IV. SIMULERINGSPAKET

1. Inledning

Den tidigare beskrivna kompilatorn och exekveringsrutinen har infogats i ett simuleringspaket. Detta styres av kommando som ges från teletypen.

Simuleringspaketets kommando är:

SYSTT (läser <system description> från teletypen)
SYSDK (läser <system description> från disken)
DISP (ger sammanställning av <identifier>:s värden på displayet)
CX0 (ändrar tillståndsvektorns initialvärde)
CPAR (ändrar <parameter>:s värde)
PLOT (läser in vilka <identifier>:s som skall plottas)
AXIS (ritar axlar på displayet)
COMPU (utför beräkningar)
SIMU (simulerar)
STOP (återvänder till monitorn)

2. Kommandon

Varje kommando består av ett inledande huvudord eventuellt följt av vissa parametrar och ord.

Syntaktisk definition av kommandona återfinns i Appendix 3.
I anslutning härtill ges nedan vissa kommentarer till varje kommando.

SYSTT

Kommando SYSTT används då man vill ange <system description> från teletypen. Varje rad kompileras för sig och skrives sedan på displayet eventuellt med tillhörande felmeddelande. Om man vill överföra <system description> till disken anges önskat <filename> efter SYSTT. Felaktiga rader ignoreras och överförs ej till disken. Avslutning göres med *END eller *CLOSE. Om *CLOSE används sätts "flagga" som omöjliggör exekvering.

SYSDK

Detta kommando används för att läsa <system description> från disken. Efter SYSDK anges <filename> för önskad fil som skall ha "extension" NON. Utskrift på displayet erhålles på samma sätt som vid kommando SYSTT. Felaktiga rader ignoreras och medför att "flagga" sätts som omöjliggör exekvering.

DISP

Används då man vill veta <identifier>:s värden. Utskrift av dessa erhålles på displayet. I utskriften ingår även tillståndsvektorns initialvärde X0.

CX0 (Change X0)

Möjliggör ändring av tillståndsvektorns initialvärde X0. Efter CX0 skrives <X0 definition> på samma sätt som i <system description>.

CPAR (Change <parameter>)

Används för att förändra <parameter>:s värde. Efter CPAR skrives <parameter definition> på samma sätt som i <system description>.

PLOT

Är ett kommando för att före beräkning eller simulering ange vilka <identifier>:s som skall plottas på displayet. Den <identifier> som skall göra avlänkning i X-led anges närmast efter PLOT. Därefter skall följa ← och mellan 1 och 10 <identifier>:s som skall avlänka i Y-led.

AXIS

Medför uppritning av axlar på displayet. Därvid anges <minimum value> och <maximum value> för respektive axel, varvid skalning för lämplig gradering göres. För att ange vilken axel angivna värden avser, sätts ett H (horizontal) eller ett V (vertical) efter AXIS. Den axel som ej berörs av kommandot erhåller samma gradering som tidigare. Om båda axlarna skall ha samma gradering som tidigare anges endast AXIS.

COMPU

Kommando COMPU medför att den exekverbara koden genomlöps <number of intervals> + 1 gånger varvid T stegas linjärt i intervallet <start value> - <stop value>. Sist i kommandot kan anges MARK, som medför att markeringar sätts på de plottade kurvorna i var tionde punkt.

SIMU

Används för att simulera dynamiska system. Den oberoende variablen T kommer då att variera i intervallet <start value> - <stop value> med initiellt inkrement (<stop value> - <start value>)/<number of intervals>. Före simulering överförs de angivna initialvärdena på tillståndsvektorn till tillståndsvektorn. Om en simulering skall fortsättas, med föregående simulerings slutvärde hos tillståndsvektorn som initialvärde, kan emellertid ovanstående förhindras genom att ange CONT i kommandot. Genom att ange MARK fås markeringar i var tionde punkt.

DHAMDI har automatisk neddelning av inkrementet om erfodras vid integreringen. Detta utnyttjas vid plottningen om kommandot innehåller EVERY, som medför att varje integrationspunkt kommer att plottas.

<upper error bound> är en felgräns som DHAMDI utnyttjar, varvid viktningen av felen hos olika tillståndsvariabler göres lika.

STOP

Medför återgång till monitorn.

Om någon parameter i kommando COMPU eller SIMU skall anta samma värde som tidigare kan denna ersättas med , (komma) eller kan kommandot eventuellt avkortas.

Om exekveringen av kommando COMPU eller SIMU skall brytas, göres detta genom att 1-ställa DATA-switch 0 (även kallad AC-switch) på PDP:ns konsol. Om därvid DATA-switch 1 är 0-ställd göres endast avbrott av exekveringen, som kan återupptagas genom att återställa DATA-switch 0. Om däremot DATA-switch 1 är 1-ställd avbryts exekveringen definitivt och nästa kommando kan ges.

3. Implementering av simuleringspaketet

För implementeringen av simuleringspaketet har skrivits, förutom de subrutiner som ingår i kompilatorn samt CALCUL, ytterligare fjorton FORTRAN-subrutiner och ett huvudprogram. Programmens inbördes relationer framgår av fig. 3.

Nedan följer en sammanställning av vad respektive program utför. De biblioteksrutiner, förutom RLINE, FAC, RABC och RFP, som anropas anges inom parantes.

SIMNON

Vid uppstart initialiseras först vissa parametrar, därefter läses kommando från teletypen. Det första ordet i kommandot avgör därvid vilken subrutin som skall anropas. Om samma huvudord önskas som i föregående kommando kan detta ersättas med , (komma).

SYSTT SYSDK DISP CX0 CPAR

Gör alla en fortsatt avkodning av respektive kommando och utför sedan vad som förväntas.

PLOT IDENT

Subrutinen IDENT beräknar adressen till <identifier>. Subrutinen PLOT avkodar kommando PLOT med hjälp av IDENT och lagrar upp adresserna till de <identifier>:s som skall plottas.

AX

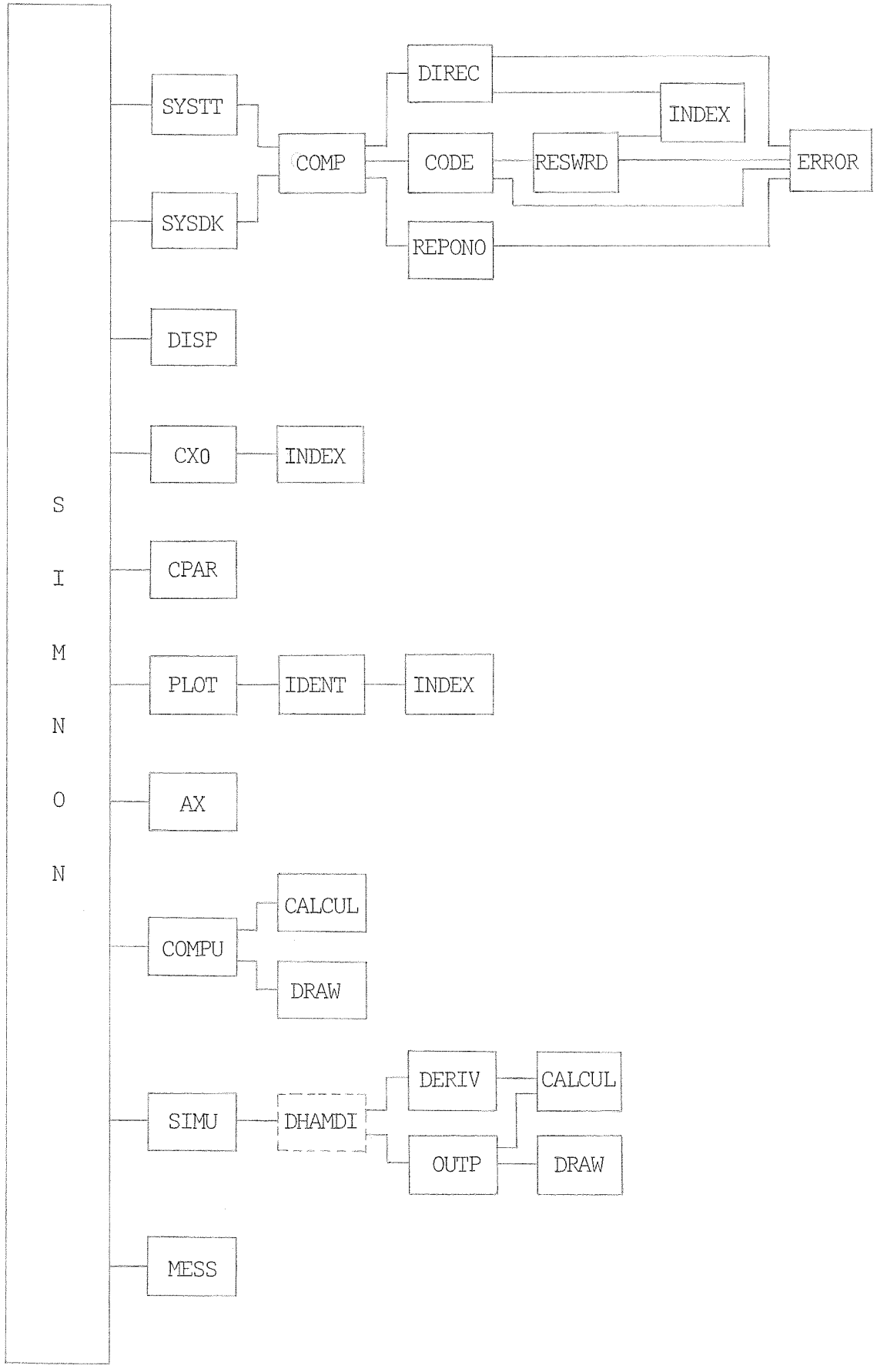
Avkodar kommando AXIS, gör eventuellt skalning och ritar axlar. (SCALE, ERASE, AXIS).

DRAW

Drar räta linjer från varje plottad <identifier>:s värde i föregående punkt till nuvarande värde. (PLOTTA, MARK).

COMPU

Avkodar kommando COMPU samt anropar CALCUL och DRAW önskat antal gånger. Därvid stegas T samt "känns" med hjälp av ISENSW på DATA-switcharna.


SIMNON
SYSTT
SYSDK
COMP
DIREC
CODE
REPONO
INDEX
RESWRD
ERROR
DISP
CX0
INDEX
CPAR
PLOT
IDENT
INDEX
AX
COMPU
CALCUL
DRAW
SIMU
DHAMDI
DERIV
OUTP
CALCUL
DRAW
MESS

Figur 3.

DERIV

Anropas av DHAMDI för beräkning av derivatornas värden. DERIV gör väsentligen ett anrop av CALCUL.

OUTP

Anropas av DHAMDI som utmatningsrutin. OUTP gör väsentligen anrop av CALCUL och DRAW samt "känner" med hjälp av ISENSW på DATA-switcharna.

SIMU

Gör avkodning av kommando SIMU och anropar DHAMDI.

MESS

Ger vissa meddelanden på teletypen.

Sammanlagda antalet FORTRAN-satser i simuleringspaketet exkl. DHAMDI (exkl. kommentarer) är ca 1650.

V. EXEMPEL PÅ ANVÄNDNING AV SIMNON

Som exempel på SIMNON:s användning demonstreras simulering av triodoscillator och styckvis linjärt system, lösning av transcendent ekvation samt användning av <own function>:s. Dessutom ges exempel på felmeddelanden.

SIMNON-Ex 1. Triodoscillator.
Triodoscillatorn beskrives i OLINJÄRA SYSTEM[1] och därifrån hämtas följande differentialekvation för gallerpotentialen u.

$$\frac{d^2u}{dt^2} + \varepsilon(1-g(u))\frac{du}{dt} + u = 0$$

Funktionen g är derivatan av triodens anodström-gallerpotential-karaktäristik och som approximation till g skall användas:

$$g(u) = \begin{cases} 2-u^2 & |u| < \sqrt{2} \\ 0 & \text{annars} \end{cases}$$

Systembeskrivningen göres på tillståndsform med $x_1=u$ och $x_2=\frac{du}{dt}$.

$$\frac{dx_1}{dt} = x_2$$

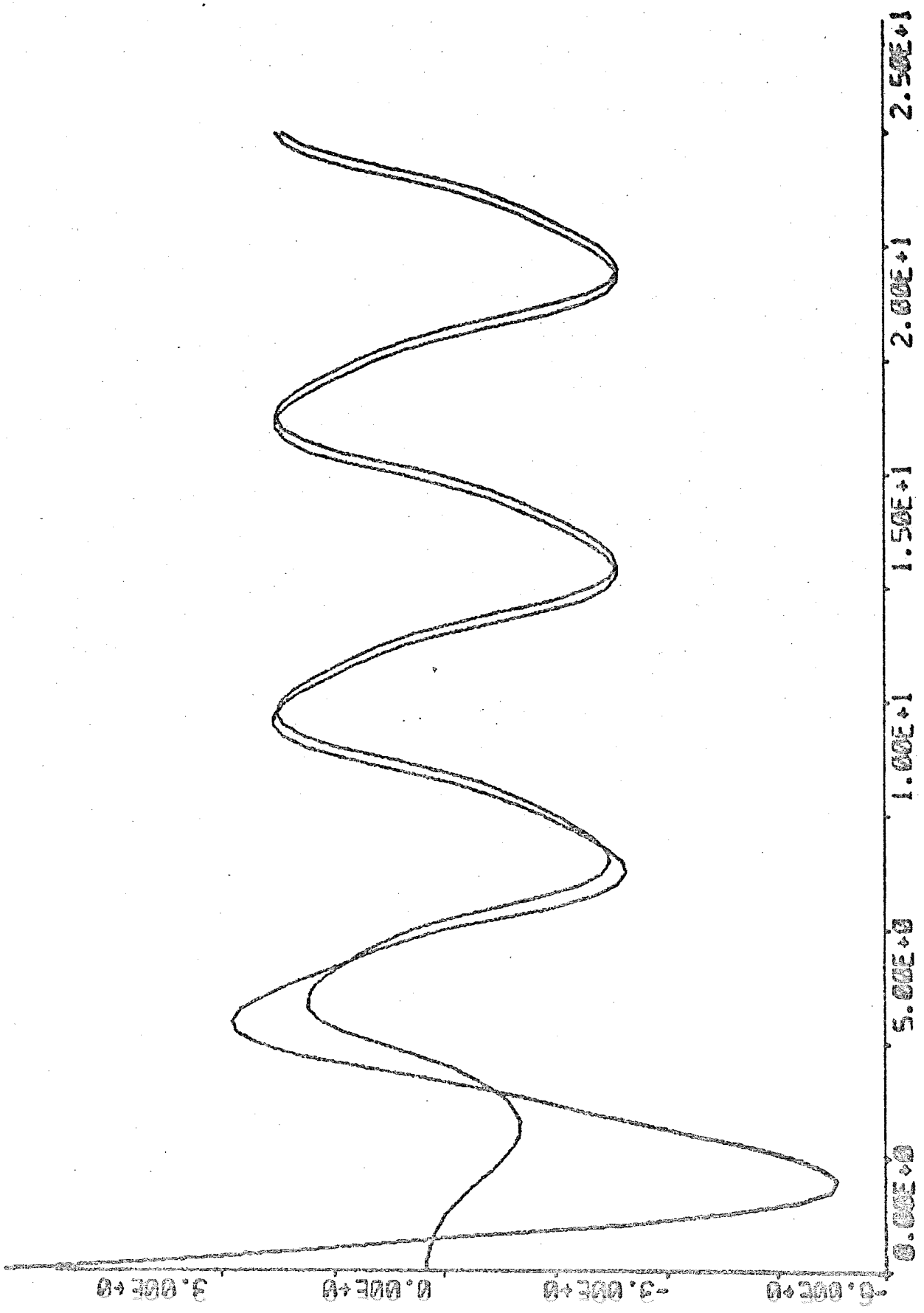
$$\frac{dx_2}{dt} = -x_1-\varepsilon(1-g(u))x_2$$

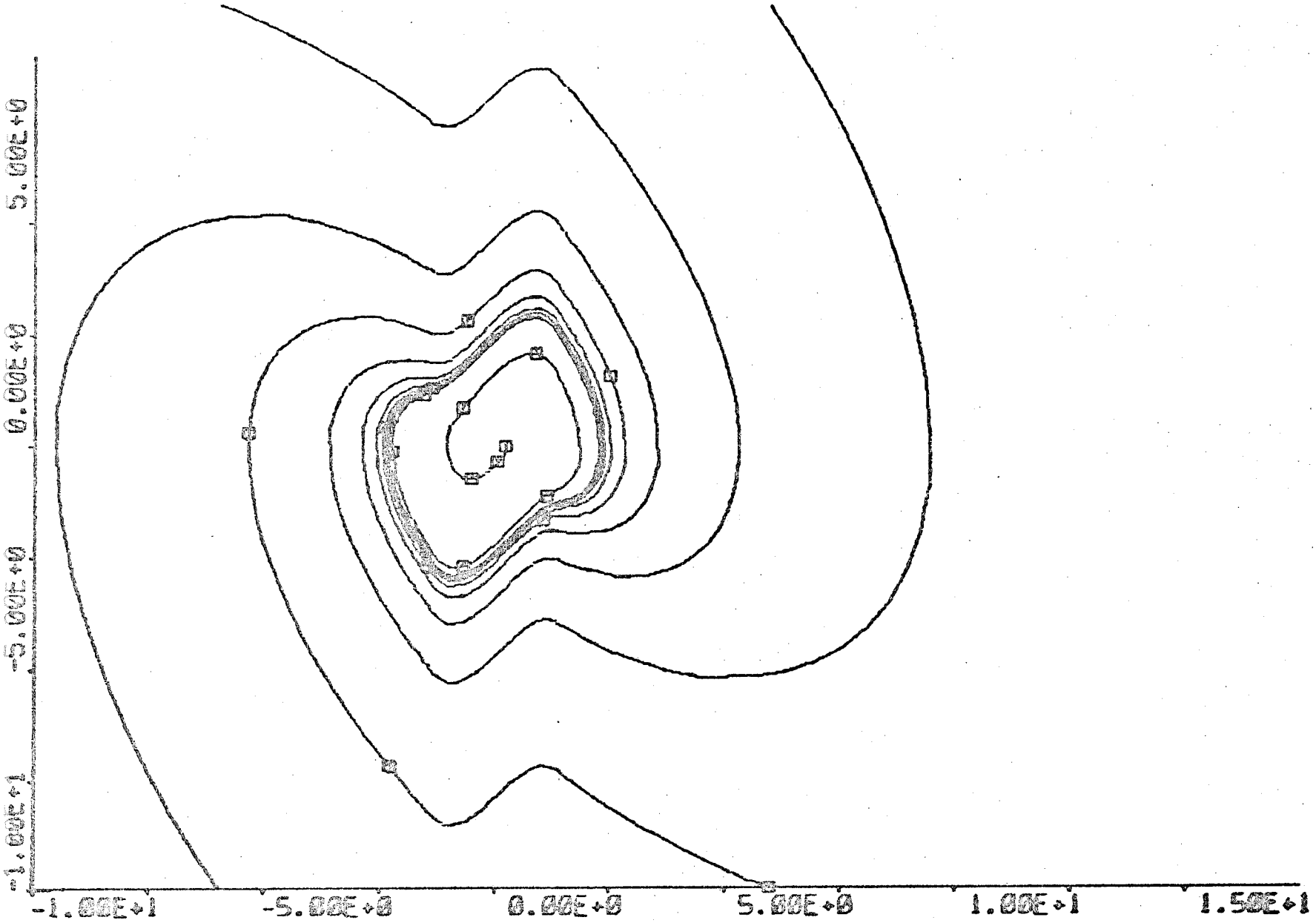
På följande sidor visas utskrift från teletypen samt bilder från displayet under simulering av triodoscillatorn med $\varepsilon = 1$. Plottning göres dels av x_1 mot t och dels av x_2 mot x_1.

1) K.J. Åström Reglerteknik Olinjära system Lund 1971

```
DE SIMMON

    SIMMON V1A
>
SYSTE TRIOD
    <SYSTEM DESCRIPTION>:

"   PICTURE 1: <SYSTEM DESCRIPTION> FOR TRIOD OSCILLATOR

G=IF ABS(X1) < LIM THEN 2-X1*X1 ELSE 0

DX1=X2
DX2=-X1-EPS*(1-G)*X2

LIM:1.414
EPS:1

*END


>"   SEE PICTURE 1
>
>PLOT T-X1   "   PICTURE 2: TRIOD OSCILLATOR
>AXIS H 0 25
>,V -5 5
>CX0 X1:0.25
>SIMU 0 25 100 1E-3
>CX0 X1:6
>,X2:-10
>SIMU
>"   SEE PICTURE 2
>
>PLOT X1-X2   "   PICTURE 3: TRIOD OSCILLATOR
>AXIS V -8 8
>,H -12 10
>CX0 X1:0.25
>,X2:0
>SIMU,10,EVERY MARK
>CX0 X1:6
>,X2:-10
>SIMU,,,MARK EVERY
>CX0 X1:-6
>SIMU,25,EVERY
>CX0 X2:10
>SIMU,,,EVERY
>CX0 X1:6
>SIMU,,,EVERY
>"   SEE PICTURE 3: THOSE FROM PICTURE 2 ARE MARKED
>
```

```
"  PICTURE 1: <SYSTEM DESCRIPTION> FOR TRIOD OSCILLATOR

G=IF ABS(X1) < LIM THEN 2-X1*X1 ELSE 0

DX1=X2
DX2=-X1-EPS*(1-G)*X2

LIM=1.414
EPS=1

*END
```


PLOT T-X1 : PICTURE 2: TRIOD OSCILLATOR
-6.00E+0
-3.00E+0
0.00E+0
3.00E+0
0.00E+0
5.00E+0
1.00E+1
1.50E+1
2.00E+1
2.50E+1


PLOT X1-X2 * PICTURE 3: TRIOD OSCILLATOR
5.00E+0
0.00E+0
-5.00E+0
-1.00E+1
-1.00E+1
-5.00E+0
0.00E+0
5.00E+0
1.00E+1
1.50E+1

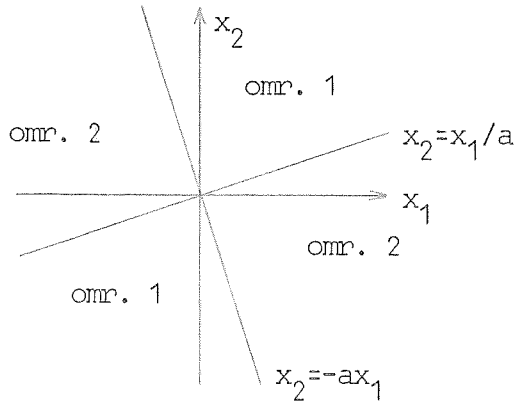
SIMNON-Ex 2. Styckvis linjärt system.

I OLINJÄRA SYSTEM behandlas ett styckvis linjärt system:

$$\frac{dx}{dt} = \begin{pmatrix} -1 & a \\ 0 & -1 \end{pmatrix} x \qquad \text{i område 1}$$

$$\frac{dx}{dt} = \begin{pmatrix} -1 & 0 \\ -a & -1 \end{pmatrix} x \qquad \text{i område 2}$$

De två områdena definieras i nedanstående figur:



I OLINJÄRA SYSTEM visas att systemet är instabilt då a är större än lösningen till ekvationen

$$a^2 = \exp(2+2a^{-2}).$$

För att lösa denna transcendenta ekvation kan SIMNON användas genom att ersätta a med tillståndsvariablen x_1 och låta x_1:s derivata beräknas som

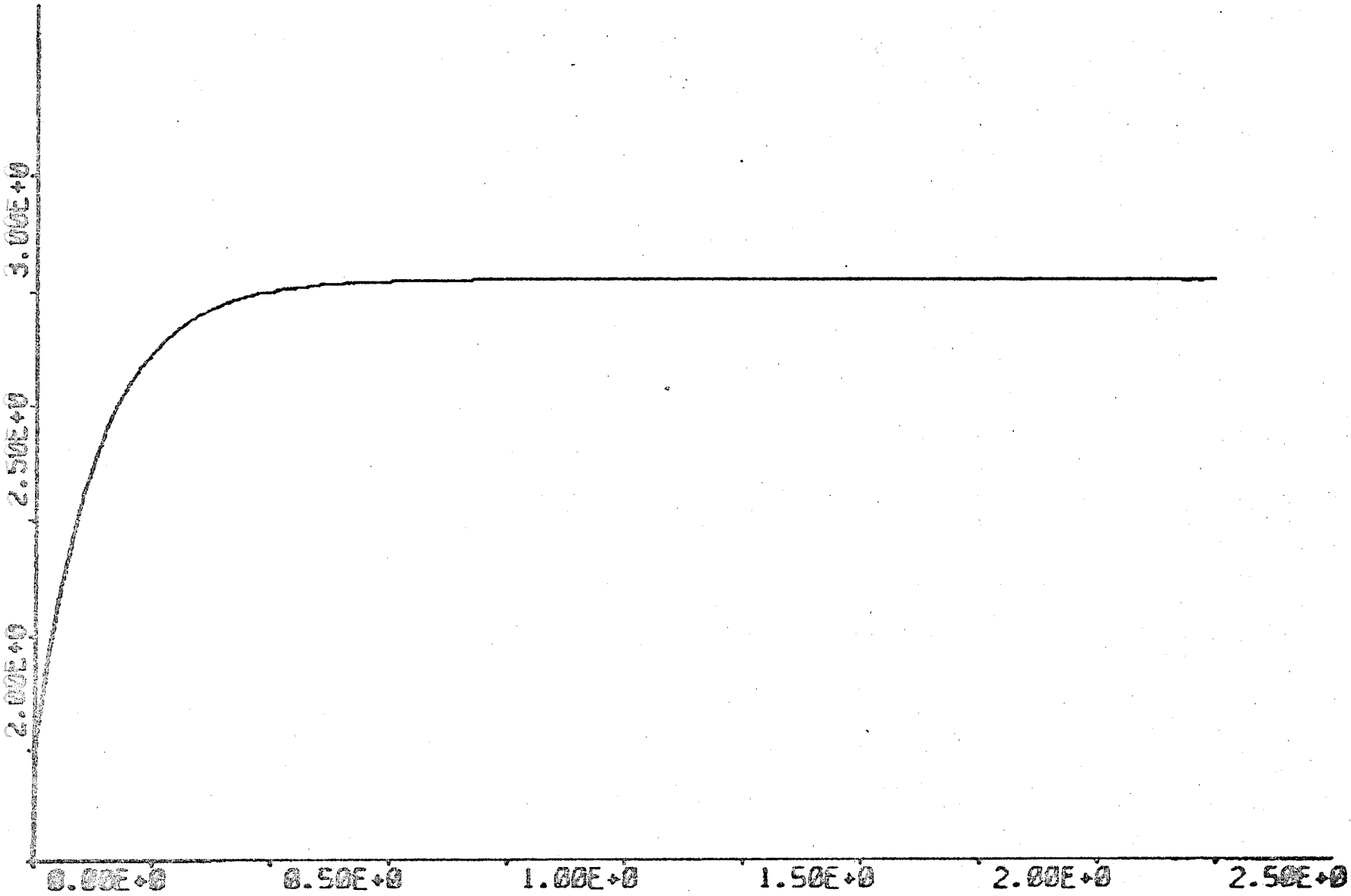
$$\frac{dx_1}{dt} = \exp(2+2x_1^{-2}) - x_1^2$$

Tecknet hos derivatan har valts så att om detta system simuleras kommer $\frac{dx}{dt}1$ att konvergera mot noll varvid x_1:s värde är lösningen till ekvationen. Eftersom den positiva roten sökes sätts x_1:s initialvärde positivt.

På följande sidor visas resultatet av exekvering.

```
>SYSTT 2A

    <SYSTEM DESCRIPTION>:


DX1=EXP(2+2*X1↑(-2))-X1↑2
X1:2
*END


>PLOT T-X1 " PICTURE 4: SOLUTION OF EXP(2+2*X1↑(-2))-X1↑2=0
>AXIS H 0 2.5
>,V 1.0 3.1
>SIMU 0 2.5 100 1E-6
>"   SEE PICTURE 4
>DISP
>"   SEE PICTURE 5: THE SOLUTION IS 1.03369
>
```

PLOT T-X1 " PICTURE 4: SOLUTION OF EXP(2*2=X1↑(-2))-X1↑2=0



PICTURE 5:

```
                    PRESENT VALUE OF <IDENTIFIER>:S

      <STATE VARIABLE>
X1      0.303089E+01

      <DERIVATIVE>
DX1    -0.476837E-06

      <TIME>
T       0.250000E+01

                    INITIAL VALUE OF <STATE VARIABLE>:S
X1      0.200000E+01
```

För att beskriva det styckvis linjära systemet är det lämpligt att införa en boolsk variabel som t.ex. är TRUE om (x_1,x_2) ligger i område 1. En sådan boolsk variabel kan uttryckas som

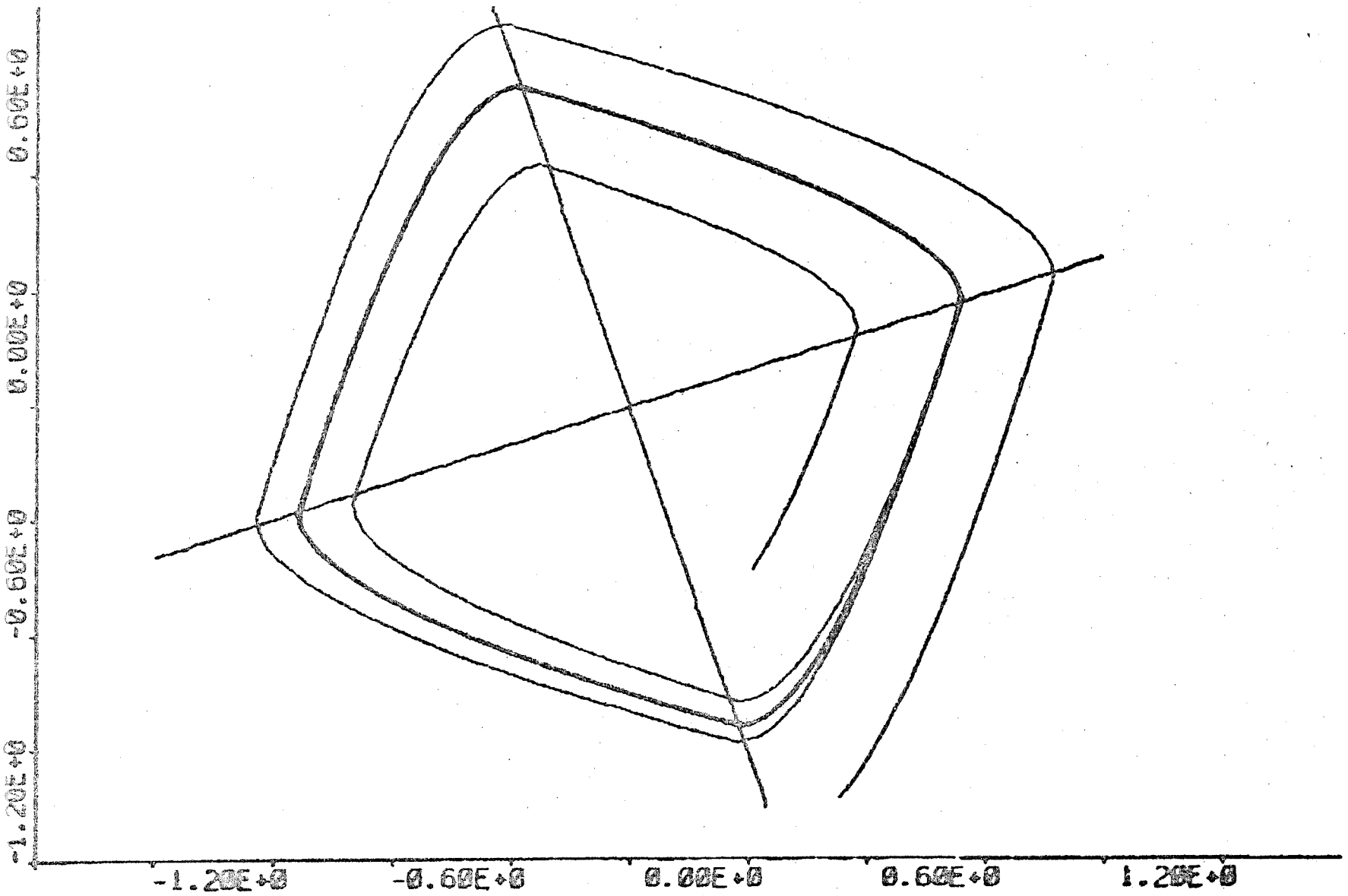
OMR1 := (X2+A*X1) * (X2-X1/A) > 0.

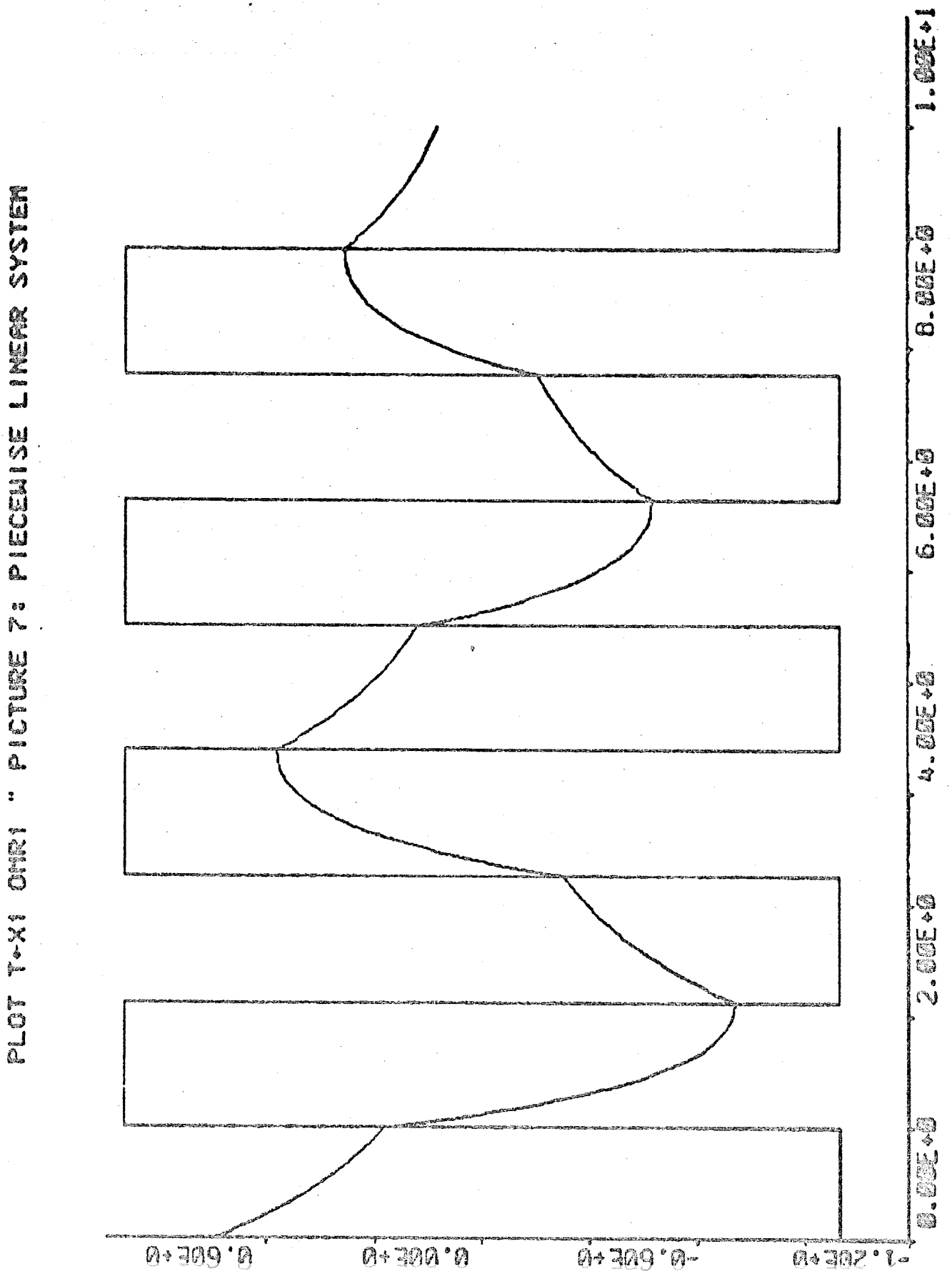
Derivatorna kan nu anges med IF-satser.

Exekvering:

```
>SYSTT 2S

    <SYSTEM DESCRIPTION>:


OMR1=(X2+A*X1)*(X2-X1/A) > 0

DX1=IF OMR1 THEN -X1+A*X2 ELSE -X1
DX2=IF OMR1 THEN -X2 ELSE -A*X1-X2
A:3.83283
X1:0.75

X2A=I/A
X2B=-I*A

*END


>PLOT X1<-X2   "  PICTURE 6: PIECEWISE LINEAR SYSTEM
>AXIS H -1.5 1.5
>,V -1 1
>SIMU 0 100 1000 1E-3
>"  THE SIMULATION MADE 22.5 WINDINGS IN THE PHASE PLANE
>CPAR A:2.8
>SIMU 0 5 100
>CPAR A:3.2
>SIMU
>
>CPAR A:3.83283
>PLOT I<-X2A   "                 BOUNDS OF AREAS: A=3.83283
>COMPU -1.2 1.2
>PLOT I<-X2B   "                          -"-
>COMPU -0.35 0.35
>"  SEE PICTURE 6
>
>PLOT I<-X1 OMR1 " PICTURE 7: PIECEWISE LINEAR SYSTEM
>AXIS H 0 12
>,V -1 1
>CPAR A:2.8
>SIMU 3 18,EVERY
>"  SEE PICTURE 7
>
```

PLOT X1-X2 " PICTURE 6: PIECEWISE LINEAR SYSTEM
PLOT T-X2A " BOUNDS OF AREAS: A=3.03089
PLOT T-X2B " -"-




PLOT T*X1 ONR1 " PICTURE 7: PIECEWISE LINEAR SYSTEM
0.000E+0
2.000E+0
4.000E+0
6.000E+0
8.000E+0
1.000E+1
-1.200E+0
-0.600E+0
0.000E+0
0.600E+0

SIMNON-Ex 3. <own function>:s.

För demonstration av <own function> har lagrats en fil med namn TREDI NON på disken.

Exekvering:

```
>SYSDX TREDI


>"  SEE PICTURE 9
>
>PLOT X←X Y   "  PI<OWN FUNCTION>:S
>AXIS H 0 30
>,V -5 50
>COMPU 0 34 34
>"  SEE PICTURE 11
>
>PLOT X←Y   "  PICTURE DIMENSION PICURE
>AXIS H -10 60
>COMPU
>"  SEE PICTURE 12
>
```

PICTURE 8:

```
*FCN1
0  0
1  40
2  40
3  0
4  0
5  14
6  54
7  54
8  14
9  14
10  23
11  16
12  0
13  18
14  31
15  40
16  54
17  36
18  31
19  31
20  40
21  54
22  36
23  31
24  18
25  0
26  14
27  23
28  18
29  18
30  23
```

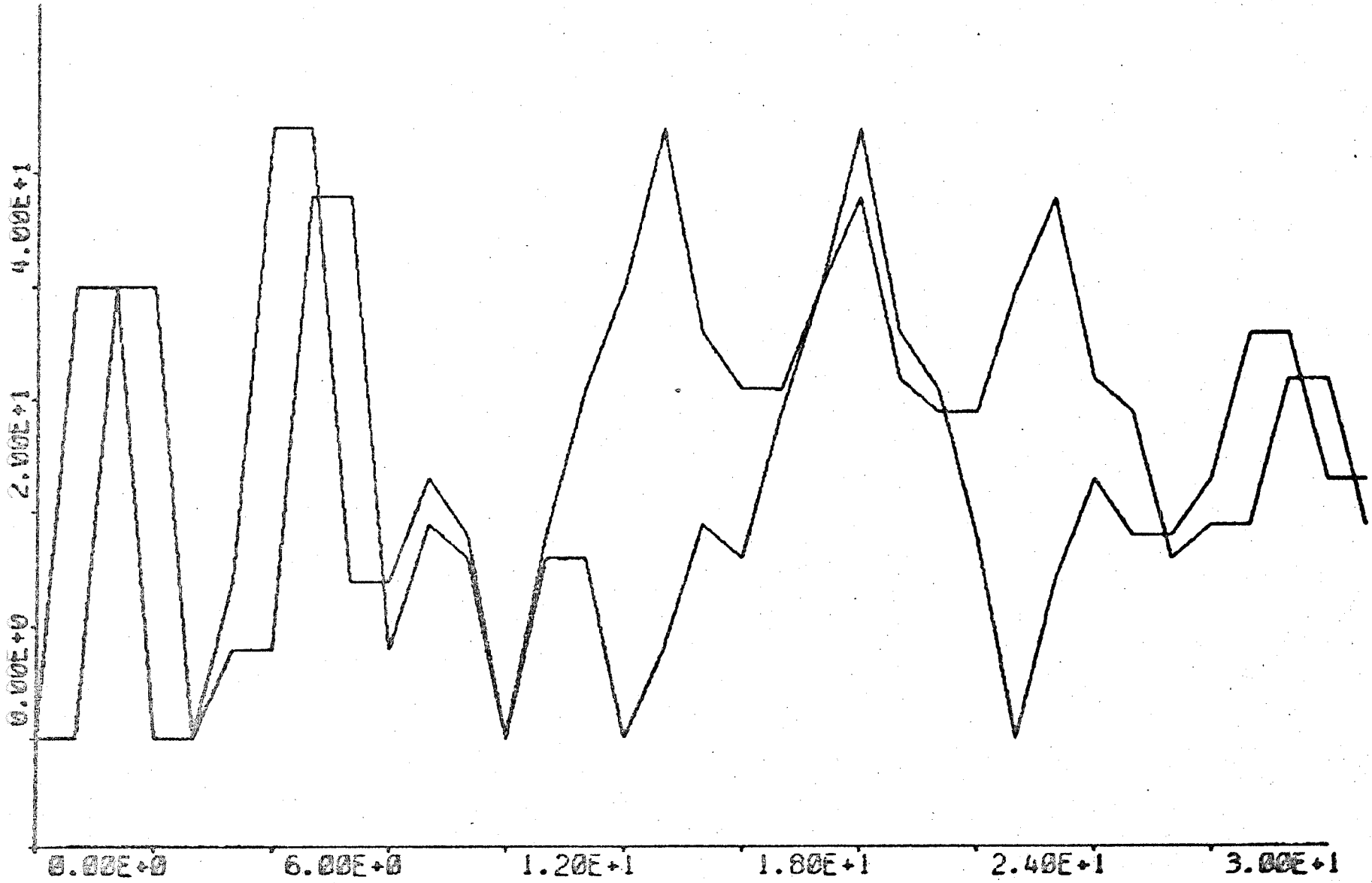
PICTURE 9:

```
31  36
32  36
33  23
34  23
*FIN
*FCN2
0  0
1  0
2  40
3  40
4  0
5  8
6  8
7  48
8  48
9  8
10  19
11  16
12  0
13  16
14  16
15  0
16  8
17  19
18  16
19  29
20  40
21  48
22  32
23  29
24  29
25  40
26  40
```

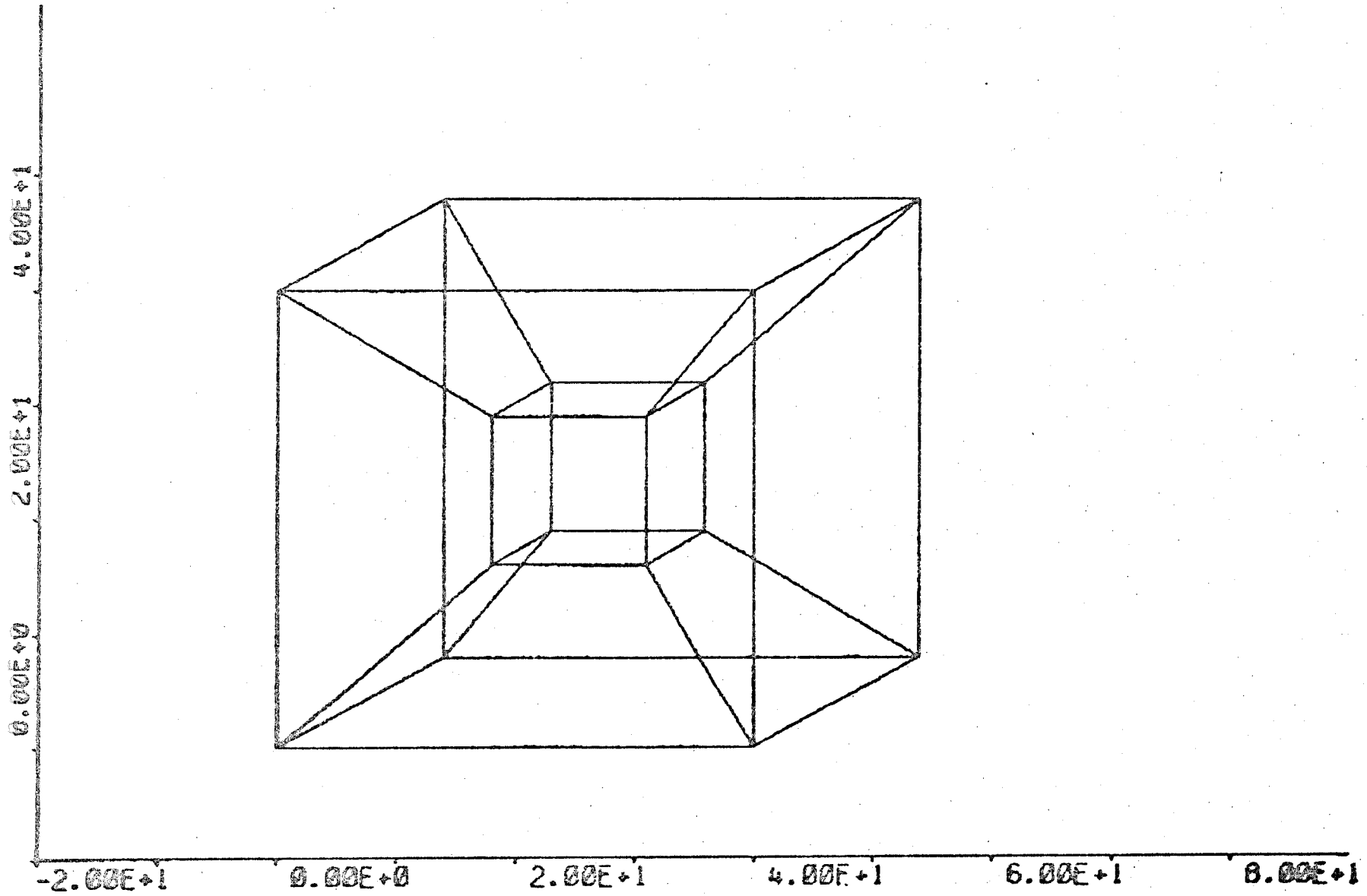
PICTURE 10:

```
27  32
28  29
29  16
30  19
31  19
32  32
33  32
34  19
*FIN
X=FCN1(T)
Y=FCN2(T)
*END
```

PLOT T←X Y " PICTURE 11: <OWN FUNCTION>:S



PLOT X-Y " PICTURE 12: THREE DIMENSION FIGURE



SIMNON-Ex 4. Felmeddelanden.

Nedan följer exempel på felmeddelanden som kan erhållas dels vid kompilering och dels vid kommandogivning.

Exekvering:

```
>SYSTT
    <SYSTEM DESCRIPTION>:

"                PICTURE 13  ERROR MESSAGES
X1=5
X4:1,5
ALFA12=X0B3B23+SIN+...
A=ATAN(X,Y)
B=SQRT(2*(X1+X2)
C=DX1+U2+Y3
F=T↑1.3
DX1=X2+FCN1(PAR×F)
*END
    PAR
    DX          2
    FCN         1
PAR:5
DX2=FCN1(SIN(SQRT(ATAN2(T*(X1-2),FCN1(F)))))
*FCN1
1.5D  3.9E5
2.0E  7.22
*FIN
*END


>"  SEE PICTURE 13
>DISP VALUES
    AFTER DISP SHOULD FOLLOW <COMMENT>
>CX3 X3:2
    INDEX OF <STATE VARIABLE> SHOULD BE IN RANGE 1 -  2
>CPAR F:3,5
    NOT <PARAMETER>: F
>PLOT X1←A
    NEVER DEFINED: A
>AXIS H 5 0
    <MAXIMUM VALUE> SHOULD BE GREATER THAN <MINIMUM VALUE>
>COMPU,,-100
    <NUMBER OF INTERVALS> SHOULD BE POSITIVE <INTEGER>
>SIMU,,,,
    NOT CORRECT CONT/MARK/EVERY/<UPPER ERROR BOUND>
>STOP NOW
    AFTER STOP SHOULD FOLLOW <COMMENT>
>STOPP
    ILLEGAL COMMAND
>STOP

STOP  000080


DOS-15 V1A
$
```

PICTURE 13: ERROR MESSAGES

```
          X1=5
  ERR1     ↑
          X4:1.5
  ERR3      ↑
          ALFA12=X000003+SIN+...
  ERR5       ↑
  ERR7                    ↑
  ERR8                     ↑
  ERR8                      ↑
  ERR8                       ↑
          A=ATAN(X,Y)
  ERR35          ↑
          B=SQRT(2*(X1+X2)
  ERR36                   ↑
          C=DX1+U2+Y3
  ERR72      ↑
  ERR74         ↑
  ERR76            ↑
F=T↑1.3
DX1=X2+FCN1(PAR=F)
          =END
  ERR96    ↑
  ERR97    ↑
  ERR98    ↑
PAR:5
DX2=FCN1(SIN(SQRT(ATAN2(T=(X1-2),FCN1(F)))))
=FCN1
1.59 3.985
2.00 7.22
=FIN
=END
```

APPENDIX 1

Syntax för simuleringsspråk

SYNTAX FOR SIMULATION-LANGUAGE

A1 <letter>:=A/B/C/D/E/F/G/H/I/J/K/L/M/N/O/P/O/R/S/T/U/V/X/Y/Z

A2 <digit>:=0/1/2/3/4/5/6/7/8/9

A3 <null>:=

A4 <name>:=<letter>{<letter>/<digit>}$_0^4$ --- <reserved word>

A5 <variable>:=<name>

A6 <parameter>:=<name>

Reserved word

B1 <index>:={<digit>}$_1^\infty$

B2 <state variable>:=X<index>

B3 <derivative>:=D<state variable>

B4 <insignal>:=U<index>

B5 <outsignal>:=Y<index>

B6 <time>:=T

B7 <own function>:=FCN<index>

B8 <one argument function>:=SIN/COS/EXP/SORT/ATAN/ABS
/ALOG/TANH/AINT

B9 <two argument function>:=ATAN2/AMOD/SIGN

B10 <boolean word>:=OR/AND/NOT/IF/THEN/ELSE

B11 <reserved word>:=<state variable>/<derivative>
/<insignal>/<outsignal>/<time>
/<own function>/<one argument function>
/<two argument function>/<boolean word>

Function designator

C1 <argument>:=<arithmetic expression>

C2 <function designator>:=
<own function>(<argument>)
/<one argument function>(<argument>)
/<two argument function>(<argument>,<argument>)

Number

D1 <unsigned integer>:=$\{$<digit>$\}_1^\infty$

D2 <integer>:={<null>/+/-}<unsigned integer>

D3 <decimal fraction>:=.<unsigned integer>

D4 <exponent part>:=E<integer>

D5 <decimal number>:=<unsigned integer>{<null>/.}
/<decimal fraction>
/<unsigned integer><decimal fraction>

D6 <unsigned number>:=
<decimal number>{<null>/<exponent part>}

D7 <number>:={<null>/+/-}<unsigned number>

Arithmetic expression

E1 <adding operator>:=+/-

E2 <multiplying operator>:=*//

E3 <identifier>:=<state variable>/<derivative>/<insignal>
/<outsignal>/<time>/<variable>/<parameter>

E4 <primary>:=<unsigned number>/<identifier>
/<function designator>/(<arithmetic expression>)

E5 <factor>:=<primary>$\{\uparrow$<primary>$\}_0^\infty$

E6 <term>:=<factor>{<multiplying operator><factor>}$_0^\infty$

E7 <simple arithmetic expression>:={null>/<adding operator>}
<term>{<adding operator><term>}$_0^\infty$

E8 <if clause>:=IF<boolean expression>THEN

E9 <arithmetic expression>:=<simple arithmetic expression>
/<if clause><simple arithmetic expression>ELSE
<arithmetic expression>

Boolean expression

F1 <relational operator>:=</=/>

F2 <relation>:=<simple arithmetic expression>
<relational operator><simple arithmetic expression>

F3 <boolean primary>:=<variable>/<parameter>/<relation>
/(<boolean expression>)

F4 <boolean secondary>:={<null>/NOT}<boolean primary>

F5 <boolean factor>:=<boolean secondary>{AND<boolean secondary>}$_0^\infty$

F6 <boolean term>:=<boolean factor>{OR<boolean factor>}$_0^\infty$

F7 <boolean expression>:=<boolean term>
/<if clause><boolean term>ELSE<boolean expression>

System description

G1 <line terminator>:=<carriage return or alt mode>

G2 <comment>:={<null>/"<the characters between " and
<line terminator>>}<line terminator>

G3 <X0 definition>:=<state variable>:<number><comment>

G4 <parameter definition>:=<parameter>:<number><comment>

G5 <own function definition>:=
*<own function>{<comment>}$_1^\infty$
{<argument value><function value>{<comment>}$_1^\infty$}$_2^\infty$
*FIN<comment>

G6 <argument value>:=<number>

G7 <function value>:=<number>

G8 <left part>:=<derivative>/<insignal>/<outsignal>/<variable>

G9 <assignment statement>:=<left part>={<arithmetic expression>
/<boolean expression>}<comment>

G10 <system description>:={<X0 definition>
/<parameter definition>/<own function definition>
/<assignment statement>/<comment>}$_0^\infty$
*END<comment>

APPENDIX 2

Felmeddelanden vid kompilering

ERROR-MESSAGES DURING COMPILATION

A. Syntactical errors (references to SYNTAX FOR SIMULATION-LANGUAGE)

0 The first character in <X0 definition>/<parameter definition> /<assignment statement> must be alfabetic (G3,G4,G9)

1 After <state variable> in <X0 definition> must follow : (G3)

2 After : must follow <number> (G3,G4)

3 After <number> in <X0 definition>/<parameter definition> must follow <comment> (G3,G4)

4 After <derivative>/<insignal>/<outsignal> in <left part> must follow = (G9)

5 <name> must not contain more than five characters (A4)

6 After <parameter> in <parameter definition> must follow : and after <variable> in <left part> must follow = (G4,G9)

7 <variable> must not appear in <parameter definition> (G4)

8 <parameter> must not appear as <left part> (G8)

9 After <own function> in <function designator>, <one argument function> and <two argument function> must follow ((C2)

10 . must be preceeded or followed by <unsigned integer> (D3,D5)

11 Allowed delimiters in <arithmetic expression> are
< > = + - * / ↑ ( ) ,

12 Unrecognized character

20 <time>/<own function>/<one argument function>/<two argument function>/<boolean word> must not appear as <left part> (G8)

30 <primary>/<boolean primary> must not be immediately preceeded by <primary>/<boolean primary>

31 NOT must be immediately preceeded by OR/AND/IF/THEN/ELSE/=/(

32 IF must be immediately preceeded by IF/ELSE/=/(

33 OR/AND/</>/=/*///↑/THEN/ELSE/)/,/<comment> must be immediately preceeded by <primary>/<boolean primary>
and +/- must be immediately preceeded by <primary>/<boolean primary>/OR/AND/NOT/</>/=/IF/THEN/ELSE/(/,

34 Too few <argument>:s have been recived for actual function (C2)

35 Too many <argument>:s are recived for actual function (C2)

36 Unrecoverable syntax error

40 After beginning * must follow <own function>/FIN/END (or CLOSE) (G5,G10)

41 After <own function> in <own function definition>, <function value>, FIN and END (and CLOSE) must follow <comment> (G5,G10)

42 <own function definition> line must begin with * , <argument value> or <comment> (G5)

43 After <argument value> must follow <function value> (G5)

44 At least two pairs of <argument value><function value> must be given in <own function definition> (G5)

B. Restrictions

50 <unsigned number> contain too many significant digits

51 It is not possible to define more <name>:s

52 Actual <parameter> has already been defined

53 Before a <variable> is used in right-part it must be defined in an earlier <assignment statement>

54 It is not possible to define more <unsigned number>:s

70 The value of <index> must not be zero

71 The value of <index> must not be greater than ten

72 Before a <derivative> is used in right-part it must be defined in an earlier <assignment statement>

73 The <derivative>:s must be consecutivly defined

74 Before an <insignal> is used in right-part it must be defined in an earlier <assignment statement>

75 The <insignal>:s must be consecutivly defined

76 Before an <outsignal> is used in right-part it must be defined in an earlier <assignment statement>

77 The <outsignal>:s must be consecutivly defined

80 It is not possible to use more RPN

90 The value of <index> in <own function definition> must not be zero

91 The value of <index> in <own function definition> must not be greater than ten

92 The <own function>:s must be consecutivly defined

93 <argument value>/<function value> contain too many significant digits

94 The <argument value>:s must increase

95 It is not possible to define more <argument value>/<function value>

96 Not all <parameter>:s have been defined

97 Not all <derivative>:s have been defined

98 Not all <own function>:s have been defined

APPENDIX 3

Syntax för kommandon

SYNTAX FOR COMMANDS

SYSTT{<null>/<filename>}<comment>

SYSDK<filename><comment>

<filename>:={<letter>/<digit>}$_1^5$

DISP<comment>

CX0<X0 definition>

CPAR<parameter definition>

PLOT<identifier><-{<identifier>}$_1^{10}$ <comment>

AXIS{<null>/{H/V}<minimum value><maximum value>}<comment>

<minimum value>:=<number>

<maximum value>:=<number>

COMPU{<start value>/,}$_i^i${<stop value>/,}$_j^j${<number of intervals>/,}$_k^k$

{{MARK}$_1^\infty$}$_1^1$<comment>

$1 \geq i \geq j \geq k \geq l \geq 0$

SIMU{<start value>/,}$_i^i${<stop value>/,}$_j^j${<number of intervals>/,}$_k^k$

{{CONT/MARK/EVERY/<upper error bound>}$_1^\infty$}$_1^1$<comment>

$1 \geq i \geq j \geq k \geq l \geq 0$

<start value>:=<number>

<stop value>:=<number>

<number of intervals>:=<integer>

<upper error bound>:=<number>

STOP<comment>