ROTSYN

ETT PROGRAM FÖR SYNTES AV LINJÄRA, TIDSINVARIANTA SYSTEM MED ROTORMETODEN

Bo Kennedy

RE - 128 juni 1973
Inst. för Reglerteknik
Lunds Tekniska Högskola

EXAMENSARBETE I REGLERINGSTEKNIK: SYNTES MED ROTORTMETODEN

utfört av BO KENNEDY
handledare JOHAN WIESLANDER
under läsåret 72/73

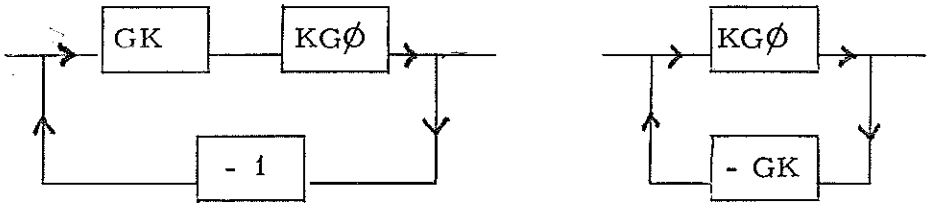
Tekniska Högskolan i Lund
inst. för regleringsteknik

INNEHÅLLSFÖRTECKNING

1. INLEDNING

Programmet ROTSYN är avsett att användas som hjälpmedel vid syntes av reglersystem.

Systemen är begränsade till klassen linjära, tidsinvarianta system, uppbyggda av ett återkopplat system och en kompenseringslänk, som kan vara serie- eller parallellkopplad. Blockschemata för de möjliga systemen ges i figur 1, där GØ och GK betecknar överföringsfunktionerna för öppna systemet respektive kompenseringslänken och K kretsförstärkningen.



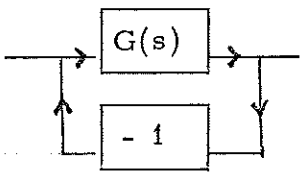
seriekompensation parallellkompensation

figur 1

Syntesen görs enligt rotortmetoden. Programmet plottar rotorten för aktuellt system på displayet. Med ledning av denna väljer användaren lämplig kompensering och låter plotta rotorten. Genom att ändra värdet på någon eller några variabler, välja ny kompenseringslänk etc. prövar man sig fram till önskat utseende på rotorten och önskade egenskaper hos systemet uppnåtts. Programmet erbjuder också möjlighet att plotta stegsvaret så att snabbhet, översläng etc. kan studeras och eventuellt kontrolleras att de uppfylla givna specifikationer.

2. LÄMPLIG METOD ATT RITA ROTORTEN?

Två metoder att upprita rotorten diskuterades vid arbetets början. Dessa redovisas i korthet nedan. För enkelhets skull antar vi att det aktuella systemet har ett utseende enligt figur 2, där $G(s) = KQ(s)/P(s)$. Den karakteristiska ekvationen är då $P(s) + KQ(s) = Ø$.



figur 2

Metod 1 Lös karakteristiska ekvationen med hjälp av biblioteksrutinen. ROT under det att K varieras med en förutbestämd

(dock ej konstant) steglängd och plotta rötterna för varje värde på K.

Denna metod skulle innebära en tidsåtgång att plotta rotorten för ett fjärde ordningens system med hundra K-värden på storleksordningen 100 sekunder.

Metod 2 Scanna över skärmen i x- och y-led och studera arg G(s) = arg Q(s) - arg P(s). För punkter på rotorten gäller arg G(s) = π + 2 kπ , k = 0,1,2,.... Genom att kontrollera när arg G(s) passerar värdet π + 2 kπ kan vi finna punkter på rotorten.

Denna metod skulle innebära att man måste räkna med komplexa tal, vilket innebär komplikationer eftersom kompilatorn på den aktuella datorn inte klarar av detta. Vad tidsåtgången beträffar kan följande anföras: Man måste löpa igenom en del av programmet några tusen gånger när man scannar displayet. Varje gång måste ARCTAN anropas minst två gånger vid beräkning av arg G(s). Eftersom ett ARCTAN-anrop tar ca 10 ms kommer man snabbt upp i tider på flera hundra sekunder bara på ARCTAN-anropen.

Redan från början valdes metod 1 för fortsatt arbete p.g.a. det enklare programmeringsarbetet och den, åtminstone i inte alltför stora system, mindre tidsåtgången. Vidare ger metoden den fördelen att man under plottningen kan se åt vilket håll kurvgrenarna rör sig. Fördelen i tidsåtgång kom emellertid att delvis försvinna då det fortsatta arbetet visade att den ursprungliga tanken på förutbestämd steglängd på K gav "fula" kurvor. Steglängden måste varieras beroende på system, vilket innebär att ROT anropas ett antal gånger utan att rötterna plottas. Å andra sidan behöver inte karakteristiska ekvationen lösas för bortåt hundra olika K-värden för att rita ett rotortdiagram p.g.a. den metod som valts, varför det bara tar i storleksordningen 10-20 sek att rita rotorten för ett system av fjärde ordningen. Möjligheten att införa en tidsfördröjning av typen e^{-Ts} avsäger man sig också genom att välja metod 1. Denna möjlighet hade funnits om metod 2 valts.

3. PROGRAMMETS UPPBYGGNAD OCH REPRESENTATIONEN AV SYSTEMET

Programmet är uppbyggt av ett huvudprogram, ROTSYN, och ett antal subrutiner. Användaren bestämmer arbetsordningen genom att på teletypen ge kommandon till ROTSYN som då anropar önskade subrutiner.

Subrutinerna kan indelas i följande avdelningar:

1. Inmatning och ändring av värden på variabler i representationerna av öppna systemet och kompenseringslänken
2. Uppritning av rotorten
3. Uppritning av stegsvaret
4. Information om aktuella värden på variabler.

Vidare finns en rutin som ger upplysning om beteckningar, tillåtna kommandon etc. främst avsedd för den ovane användaren.

I kommunikationen användare-program representeras öppna systemet och kompenseringslänken med sina överföringsfunktioner, betecknade GØ resp. GK. Dessa kan skrivas på två sätt: täljare och nämnare skrivna i polynomform eller faktoruppdelade. Dessa olika sätt betecknas AB- respektive PZ-version.

Följande beteckningar på variablerna i GØ och GK användes:

$$G\emptyset(s) = \frac{s^{M\emptyset} + B\emptyset(1) * s^{M\emptyset-1} + \ldots + B\emptyset(M\emptyset)}{s^{N\emptyset} + A\emptyset(1) * s^{N\emptyset-1} + \ldots + A\emptyset(N\emptyset)} K\emptyset \qquad \text{(AB-version)}$$

$$= K\emptyset * \frac{(s-ZR\emptyset(1) - I*ZI\emptyset(1)) * \ldots * (s-ZR\emptyset(M\emptyset) - I*ZI\emptyset(M\emptyset))}{(s-PR\emptyset(1) - I*PI\emptyset(1)) * \ldots * (s-PR\emptyset(N\emptyset) - I*PI\emptyset(N\emptyset))} \qquad \text{(PZ-version)}$$

I betecknar en imaginär etta dvs. $I^2 = -1$

GK(s) skrives analogt med Ø ersatt med K dvs. MØ, BØ etc. ersatt med MK, BK etc. Både AB- och PZ-versionerna finns hela tiden lagrade även om enbart ena versionen matats in.

Högsta tillåtna gradtal för alla täljare och nämnare i GØ och GK är 1Ø dvs. MØ, NØ, MK och NK måste alla vara mindre än eller lika med 1Ø. Samtidigt måste gradtalen i nämnare och täljare på totala systemets överföringsfunktion vara högst 1Ø. Detta innebär att följande villkor måste vara uppfyllda: $N\emptyset + NK \leq 1\emptyset$ och $M\emptyset + MK \leq 1\emptyset$. Vid parallellkompensation måste dessutom gälla att $M\emptyset + NK \leq 1\emptyset$. Dessa begränsningar beror på att den i programmet använda biblioteksrutinen ROT klarar av att lösa ekvationer av högst gradtalet 1Ø.

4. BESKRIVNING AV PROGRAMMET

4.1 ROTSYN

Huvudprogrammet ROTSYN är programmets sammanhållande länk. Genom det styr användaren exekveringen. Han ger kommandon på teletypen, som tolkas av ROTSYN. ROTSYN anropar sedan avsedda subrutiner. När dessa exekverats är ROTSYN klart för nästa kommando.

ROTSYN ger, när exekveringen av programmet börjar, en utskrift, främst avsedd för den ovane användaren, som talar om att det finns ett kommando, GUIDE, som ger en utskrift där beteckningar och möjliga kommandon förklaras.

ROTSYN är klart att ta emot ett kommando när det på teletypen givit utskriften:

YOUR COMMAND:

>

Om det givna kommandot är felaktigt sker felutskrift och nytt kommando efterfrågas.

Om användaren begär att rotorten eller stegsvaret ska ritas och inte både GØ och GK har blivit inmatade skrivs detta ut, varefter nytt kommando efterfrågas, utan att respektive rutiner anropats. Således måste även GK=1, dvs. ingen kompensering, matas in.

Vilka kommandon har då användaren att välja på? Nedan ges en uppställning av tillgängliga kommandon och en kort beskrivning av deras innebörd. Inom parentes anges de subrutiner som ROTSYN anropar vid respektive kommando (I kapitlet "User's guide" ges en mer detaljerad bild av följden av ett visst kommando, vilka värden som kommer att efterfrågas etc.)

a) GUIDE ger en utskrift som förklarar de beteckningar, som användes i kommunikationer mellan användare och program, samt en kort översikt över kommandoorden.
   (GUIDE)

b) INAB är en inputrutin för överföringsfunktionerna givna i AB-versionen. PZ-versionen beräknas och lagras.
   (INAB)

c) INPZ är motsvarande inputrutin för överföringsfunktionerna i PZ-version. AB-versionen beräknas och lagras.
   (INPZ)

d) CHAB är en rutin för att ändra något variabelvärde i överföringsfunktionernas AB-version. Erforderlig ändring i PZ-versionen utförs automatiskt.
(CHAB)

e) CHPZ är motsvarande rutin för PZ-versionen. Erforderlig ändring i AB-versionen utförs automatiskt.
(CHPZ)

f) LISTØ ger en listning på teletypen av aktuella värden på GØ, både AB- och PZ-version.
(LISTØ)

g) LISTK är motsvarigheten till LISTØ men avser GK..
(LISTK)

h) ROTORT ritar rotorten för aktuellt system. Möjlighet finns att få utmärkt speciella värden på förstärkningen.
(FB, ROTIO, ROTAX, ROTRIT, INGAIN, PLGAIN)

i) STEGSVAR ritar stegsvaret för aktuellt system.
(STDATA, FB, MATRIX, SAMPLA, STEGSV)

j) END avslutar exekveringen.

N.B. Endast de fem första tecknen i kommandona är signifikanta. Om stegsvaret ska ritas räcker alltså kommandot STEGS.

4:2 GUIDE

Subrutinen GUIDE är en rutin avsedd främst för den ovane användaren. Rutinen ger en utskrift som talar om de beteckningar på GØ- och GK-variablerna, som användes i kommunikationen mellan användare och program. Vidare ges en kortfattad översikt över kommandona.

Utmatningsenhet bestämmer användaren själv genom att svara på frågan:

WRITE WANTED OUTPUT DEVISE: TT, LP OR VP
>

Svaret ska utgöras av någon av dessa tre förkortningar, vilka som vanligt betyder teletype, radskrivare respektive display.

Bilaga 4.1 visar hur utskriften ser ut.

4.3 INAB

Subrutinen INAB är en inputrutin för överföringsfunktioner givna i AB-version. Användaren matar här in önskade värden på variablerna i GØ eller GK. Inputenhet är teletypen.

Inmatningen sker enligt följande. Först frågar rutinen om inmatningen avser GØ eller GK:

GØ OR GK

>

Användaren svarar med GØ eller GK. Om svaret är felaktigt fås felutskrift och frågan upprepas. Om man inte svarar utan gör ALT MODE eller CARRIAGE RETURN direkt, sker återhopp till huvudprogrammet.

När rutinen fått svar på ovanstående frågar den efter värdena på KØ, MØ, NØ eller KK, MK, NK, beroende på det tidigare svaret. Exempelvis

WRITE KØ, MØ, NØ

#

När dessa värden matats in frågar rutinen efter värdena på BØ-, AØ-alternativt BK-, AK-vektorerna. De vektorer för vilka gradtalet Ø matats in efterfrågas ej. Exempel

WRITE AK(1), ..., AK(N)

#

eller

WRITE BØ(1), ... , BØ(M), AØ(1), ... , AØ(N)

#

Under inmatningen, som sker oformatterad lagras värdena i hjälpvariabler, för att, när denna är avslutad, överföras till respektive variabler, varvid värdena på PZ-versionens variabler beräknas. Därefter sker återhopp till huvudprogrammet.

Kommunikationen med huvudprogrammet sker genom COMMON-fälten OPEN och COMP, vilka innehåller värdena på alla variabler i både AB- och PZ-version för GØ respektive GK.

4.4 INPZ

Subrutinen INPZ är motsvarigheten i PZ-version till INAB. Rutinerna är till en början analoga men i stället för BØ-, AØ- respektive BK-, AK-vektorer ska man mata in nollställen och poler. Rutinen frågas sålunda exempelvis:

WRITE ZRK(1), ZIK(1), ... , ZRK(M), ZIK(M)
ONLY ONE ROOT/LINE AND COMPLEX-CONJUGATED ALLOWED

#

Värdena läses och lagras i hjälpvariabler och en kontroll göres att inmatade värden är komplexkonjugerade. Om så inte är fallet fås felutskrift och nytt försök, annars överförs värdena till respektive variabler, AB-versionen beräknas, och följande utskrift erhålles:

WRITE PRK(1), PIK(1), ... , PRK(N), PIK(N)

#

Förfaringssättet här är helt analogt med ovanstående.

Observera alltså att för reella poler och nollställen måste imaginärdelen = Ø också matas in.

Om något av de tidigare inmatade gradtalen skulle vara Ø bortfaller naturligtvis motsvarande del av ovanstående förfarande.

Kommunikation med huvudprogrammet och format vid inmatningen är samma som för INAB.

4.5 CHAB

Subrutingen CHAB är avsedd att användas när man vill ändra på enstaka värden i GØ eller GK, men vill slippa mata in hela funktionerna.

Det finns inte möjlighet att ändra täljarnas eller nämnarnas gradtal på detta sätt, utan i så fall måste man utnyttja INAB eller INPZ.

Principen vid ändringen är följande: Först anges namnet på den variabel, vars värde man vill förändra, därefter index (om variabeln är ett vektorelement) och sist det nya värdet. Parenteser, likhetstecken, radbyte o.dyl. kan inplaceras efter eget önskemål, de har ingen betydelse.

Exempel på ändringar

CHAB AØ(2) = 1.3

CHAB AØ 2

1.3

CHAB AØ

2 1.3

CHAB

WRITE VARIABLE, INDEX (IF VECTOR) AND NEW VALVE

> AØ 2 1.3

Dessa ger alla resultatet AØ(2) = 1.3

Om felaktigt namn eller ett index utanför aktuella gränser användes fås felutskrift och möjlighet till nytt försök ges.

Möjlighet till uthopp utan ändring finns genom antingen ALT MODE eller CARRIAGE RETURN direkt efter CHAB eller medvetet felaktig inmatning och motsvarande åtgärd direkt efter felutskriften.

Ändringen i respektive PZ-version görs automatiskt av rutinen, varefter återhopp sker till huvudprogrammet. Kommunikationen med huvudprogrammet sker genom COMMON-fälten OPEN och COMP samt ICHAB som innehåller numret på första positionen efter kommandoordet CHAB samt hela kommandoordsraden lagrad i en vektor. Detta möjliggör ändringar enligt de tre första exemplen ovan.

4.6 CHPZ

Subrutinen CHPZ är PZ-versionens motsvarighet till AB-versionens CHAB. Inmatning och kommunikation är helt analoga med motsvarande delar i CHAB, dock måste givetvis PZ-versionens variabelbeteckningar användas.

Eftersom det i detta fall rör sig om komplexa tal tillkommer emellertid vissa problem. Tre fall kan urskiljas:

a) Ändring av imaginärdel. Då måste man leta fram den komplexkonjugerade roten och ändra dess imaginärdel på samma sätt för att AB-versionen ska förbli reell.
b) Ändring av realdel. Även då måste man leta fram den komplexkonjugerade roten och av samma anledning som ovan ändra också dess realdel.
c) Reella rötter föranleder ingen speciell åtgärd.

Sammanfattningsvis måste man alltså ta reda på typen av rot för att kunna vidta riktiga åtgärder. Om man försöker göra några otillåtna ändringar, exempelvis ändra imaginärdelen på en reel rot, sker felutskrift.

När den avsedda ändringen genomförts, beräknas motsvarande ändringar i AB-versioen och återhopp sker till huvudprogrammet.

4.7 LISTØ

Subrutinen LISTØ listar på teletypen samtliga värden på variablerna i GØ, alltså både AB- och PZ-versionerna.

Förutom, som kontroll av inmatade värden, kan rutinen användas för att ta reda på vilka index poler och nollställen har, om inmatning har skett i AB-version och vice versa.

Kommunikation med huvudprogrammet sker genom COMMON-fältet OPEN.

4.8 LISTK

Subrutinen LISTK är GK:s exakta motsvarighet till LISTØ. COMMON-fältet är givetvis i detta fall COMP:

4.9 FB

Subrutinen FB är den rutin som utifrån GØ och GK beräknar överföringsfunktionen för hela systemet, dvs. med återkoppling och kompensering.

Beräkningen sker genom att multiplicera ihop täljarna och nämnarna i GØ och GK enligt olika alternativ. Vilket alternativ som ska användas beror på typ av kompensering, och avsikt. Första variabeln i COMMON-fältet STDAT bestämmer alternativet.

Tre alternativ finns

a) endast överföringsfunktionens nämnare beräknas. För att rita rotorten räcker detta.

b) hela överföringsfunktioen beräknas och kompenseringen sker i serie

c) hela överföringsfunktionen beräknas och kompenseringen sker parallellt.

Något av de två sista alternativen användes när stegsvaret ska bestämmas.

Den beräknade överföringsfunktionen lagras i COMMON-fälten TAELJ och NAEMN och skrivs enligt följande:

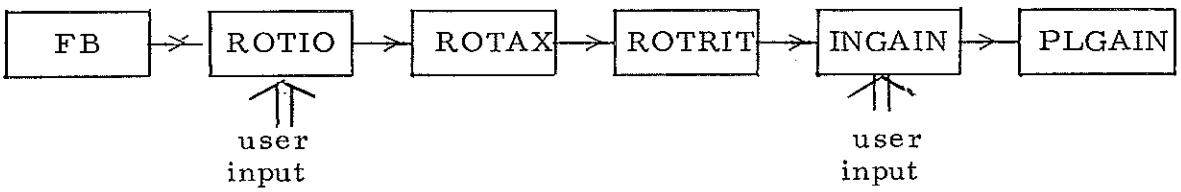
$$G(s) = \frac{E(1)*s^{M} + \ldots + E(M+1)}{(C(1)+K*D(1))*s^{N} + \ldots + (C(N+1)+K*D(N+1))}$$

där K betecknar förstärkningen dvs. den variabel m.a.p. vilken rotorten ritas.

Om systemet blir för stort, dvs. M eller N större än 1Ø sker felutskrift och återhopp (de gamla värdena förstörda). Detta innebär följande begränsningar: MØ + MK $\leq$ 1Ø, NØ + NK $\leq$ 1Ø samt vid parallellkompensering även MØ + NK $\leq$ 1Ø.

COMMON-fälten OPEN, COMP och STDAT förser rutinen med behövliga data.

Programmet innehåller en avdelning som handhar beräkningen och plottningen av rotorten. Denna avdelning består av de nedan beskrivna rutinerna ROTIO, ROTAX, ROTRIT, INGAIN, PLGAIN samt rutinen FB, som förser avdelningen med indata avseende systemets överföringsfunktion. Uppbyggnaden framgår av nedanstående figur.



figur 3

4.10 ROTIO

Subrutinen ROTIO är den rutin i vilken användaren matar in minsta och största värde på förstärkningen som rotorten ska plottas för. Rötterna för dessa värden beräknas, och med ledning av dessa beräknas data för axlarna i rotortdiagrammet såsom ändvärden, skal-

indelning. Samma skalindelning användes för båda axlarna.

Indata för att beräkna rötterna fås genom COMMON-fältet NAEMN och utdata lämnas genom AXDATA.

4.11 ROTAX

Subrutinen ROTAX ritar axlarna till rotortdiagrammet med utgångspunkt från värdena från ROTIO som fås genom COMMON-fältet AXDATA.

4.12 ROTRIT

Subrutinen ROTRIT är den rutin som sköter beräkning och plottning av rotorten. Axlarna är givna från ROTAX och de värden som förstärkningen ska variera mellan har matats in av användaren i ROTIO.

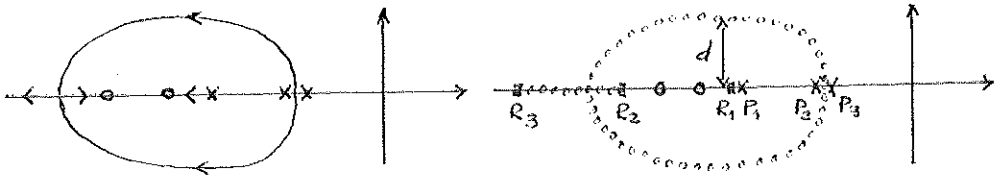
Principiellt fungerar ROTRIT enligt följande. Beräkna och plotta rötterna till karakteristiska ekvationen för förstärkningens minsta värde. Välj en steglängd (Ø.1) på förstärkningen och öka den med detta värde. Beräkna rötterna och bestäm hur långt respektive rot flyttat sig. Korrigera steglängden tills den längsta förflyttningen för en uppsättning rötter håller sig inom vissa gränser från ett idealvärde. Plotta de rötter som ligger nära idealvärdet. Spara värdet på dom rötter som plottats. Öka förstärkningen med det tidigare värdet på steglängden. Beräkna de nya rötterna och förflyttningarna från de senast plottade punkterna. Upprepa proceduren ovan. När förstärkningen överstiger det tidigare inmatade maxvärdet sker återhopp till huvudprogrammet.

Innan plottningen sker måste rötterna räknas om till koordinater. Detta sker med hjälp av de värden på skalindelning etc. som användes av ROTAX för att rita axlarna.

Eftersom det inte är säkert att rötterna under sina förflyttningar i rotortdiagrammet hela tiden ges samma index av rutinen ROT, som beräknar dem, uppstår problem i samband med att bestämma dessa förflyttningar. Följande metod har då valts. Spara de hundra senast plottade punkterna i särskilda vektorer. Från varje "ny" rot beräknas avstånden till samtliga dessa hundra punkter. Rätt förflyttningsväg väljes som det kortaste avståndet.

N. B. Det kan föreligga en liten risk att denna rutin hänger sig. Antag att vi har en rotort med flera grenar där en gren växer mycket långsamt jämfört med de övriga. Eventuellt kan då den senast plottade punkten på den långsamma grenen falla bort ur "vektorerna för gamal punkter" och ersättas med punkter på de snabba grenarna. När avstånden från den aktuella roten på den långsamma grenen till

de hundra gamla punkterna beräknas kommer kanske dessa därför att bli mycket större än tillåtet, eftersom avstånden beräknas till punkter på andra kurvgrenar, varför steglängden minskas. Den kan emellertid aldrig minskas så mycket att något avstånd blir tillräckligt litet. Nedanstående exempel belyser resonemanget ovan.



systemets rotort figur 4 ROTSYN's rotort

$P_{1, 2, 3}$ är överföringsfunktionens poler. $R_{1, 2, 3}$ är rötterna för aktuellt värde på förstärkningen. Det avstånd vi vill veta är R_1P_1, men rutinen väljer som rätt avstånd sträckan d, vilken aldrig kan fås tillräckligt liten hur mycket än steglängden minskas. Skulle detta fall uppstå kan man avhjälpa felet genom att gå in i rutinen och öka storleken på vektorerna där dom plottade punkterna sparas dvs. IR1, II1.

De hittills beskrivna rutinerna ROTIO, ROTAX och ROTRIT i programmets rotortavdelning ger ingen möjlighet att få utmärkt var rötterna för speciella värden på förstärkningen ligger. Denna möjlighet erbjuder rutinerna INGAIN och PLGAIN.

4.13 INGAIN

Subrutinen INGAIN är den rutin i vilken användaren matar in de värden på förstärkningen som han vill ha speciellt utmärkta. Högst tio värden kan matas in. Värdena lagras i COMMON-fältet GAINS i form av antal och värden.

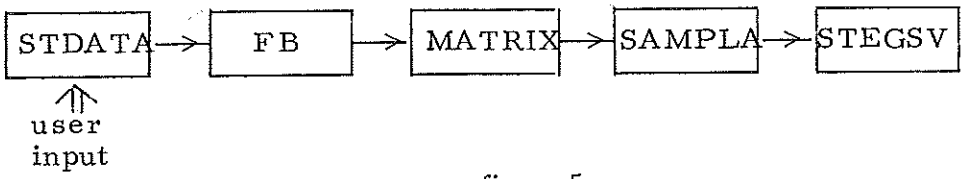
4.14 PLGAIN

Subrutinen PLGAIN beräknar och plottar rötterna för de värden på förstärkningen som matats in i rutinen INGAIN och som erhålles genom COMMON-fältet GAINS.

Värdet på förstärkningen för respektive rötter skrivs ut strax ovanför och till vänster om rötterna. Vid komplexkonjugerade rötter sker dock ingen utskrift för den rot som har negativ imaginärdel.

N. B. Om två rötter ligger nära varandra är det möjligt att värdena på förstärkningen skrivs över varandra.

Subrutinerna STDATA, MATRIX, SAMPLA och STEGSV bildar tillsammans med FB den del av programmet som handhar beräkning och plottning av systemets stegsvar. Uppbyggnaden framgår av nedanstående figur.



figur 5

4.15 STDATA

Subrutinen STDATA är den rutin där användaren förser programmet med data, nödvändiga för beräkning av stegsvaret. Först frågas vilken förstärkning som stegsvaret ska ritas för:

WRITE WANTED GAIN

#

Därefter frågar rutinen efter typ av kompensering, serie eller parallell:

SER. OR PAR. COMP.

>

Användaren svarar med någon av bokstäverna S eller P.

Observera att om GK = 1, dvs. ingen kompensering, är det ingen skillnad på serie- och parallellfallet, varför denna del av STDATA då hoppas förbi.

De inmatade värdena lagras i COMMON-fältet STDAT.

4.16 MATRIX

Subrutinen MATRIX är den rutin som beräknar hur lång tid det kan vara lämpligt att följa stegsvaret. Vidare beräknas systemets representation på matrisform.

Den ovan nämnda tiden beräknas enligt följande. Lös karakteristiska ekvationen till systemet. Ur dess rötter kan stegsvarets principiella utseende utläsas. Om stabilt, monotont stegsvar följ det tills stationära felet tillräckligt litet. Om oscillativt, stabilt eller instabilt, följ under fem perioder. Om instabilt monotont följ så att utsignalen hinner blir större än 1.5.

Eftersom stegsvarets utseende utläses ur största real- och imaginärdel hos rötterna, utan hänsyn till övriga, stämmer inte stegsvarets verkliga utseende med det förmodade, varför den beräknade tiden kan tyckas inte uppfylla ovanstående resonemang i alla fall.

Samplingstiden bestämmes till 1/1ØØ av den ovan bestämda tiden.

Ur totala systemets överföringsfunktion, given i COMMON-fälten TAELJ och NAEMN, bestämmes systemmatriserna A och B på observerbar kanonisk form. De lagras i COMMON-fältet MATR.

4.17 SAMPLA

Subrutinen SAMPLA samplar systemet, dvs. beräknar systemmatriserna A och B i tidsdiskret form, utgående från de kontinuerliga matriser, som beräknades i subrutinen MATRIX.

De beräknade samplade matriserna A och B lagras på samma plats som de kontinuerliga, dvs. i COMMON-fältet MATR.

4.18 STEGSV

Subrutinen STEGSV itererar fram stegsvaret med hjälp av de samplade systemmatriserna och samplingstiden, som är beräknade tidigare och finns lagrade i COMMON-fältet MATR.

Först ritas axlarna till diagrammet och en linje som markerar utsignalen = 1. Därefter börjar iterationen. Värdet på utsignalen omräknas till koordinater och plottas efterhand. Iterationen avbryts efter den tid som MATRIX beräknat att stegsvaret skulle följas och återhopp till huvudprogrammet görs.

5. EXEMPEL

5.1 Antag att vi har det öppna systemet $G\emptyset = 1/s^2$ återkopplat med -1. Detta system matas in genom de två första kommandona på bilaga 5.1 (Obs. GK = 1 måste också matas in.) Tredje och fjärde kommandona ger rotorten och stegsvaret (förstärkningen = 1) för systemet. Dessa finns återgivna på bilagorna 5.3 och 5.4.

Antag nu att vi vill införa kompensering i systemet för att stabilisera detta, och få det att uppfylla vissa specifikationer i tidsplanet.

Vi väljer seriekompensering och kompenseringslänken GK=s+1. Inmatningen av denna görs i femte kommandot. De två efterföljande ger rotorten och stegsvaret (förstärkningen = 2) för systemet. Se bilagorna 5.5 respektive 5.6. Antag att specifikationerna har uppfyllts.

GK=s+1 är emellertid svår att realisera eftersom den innehåller en deriverande del. Om vi i stället väljer GK=1$\emptyset$(s+1)/(s+1$\emptyset$) är denna lättare att realisera och kan antagas ha ungefär samma egenskaper som GK=s+1 åtminstone för låga frekvenser.

I andra kommandot på bilaga 5.2 matas GK=1$\emptyset$(s+1)/(s+1$\emptyset$) in och därefter ritas rotorten och stegsvaret (förstärkningen = 2) som återges på bilagorna 5.7 respektive 5.8. Som synes överens-

stämmer stegsvaret mycket väl med dito för GK=s+1, varför GK=1Ø(s+1)/(s+1Ø) och förstärkningen = 2 kan vara en lämplig lösning på vårt problem.

5.2 I detta fall är det öppna systemet GØ=1/(s^3+3s^2+2s) återkopplat med - 1. Systemet matas in i de två första kommandona på bilaga 5.9. Dess rotort och stegsvar (förstärkningen = 2) ses på bilagorna 5.11 respektive 5.12 och får genom tredje och fjärde kommandona på bilaga 5.9.

Detta system uppfyller givna specifikationer i tidsplanet men har ett hastighetsfel som man genom kompensering vill minska. Kompenseringslänken GK=(s+Ø.1)/s kopplad i serie ger systemet ett hastighetsfel av acceptabel storlek. GK matas in genom femte kommandot på bilaga 5.9 och det sjätte ger rotorten enligt bilaga 5.13.

Kommandona på bilaga 5.10 ger systemets stegsvar för förstärkningen = 2, respektive = 1. På bilagorna 5.14 respektive 5.15 återges dessa. Ur den första av dessa kan vi se att stegsvaret för förstärkningen = 2 ej ändrat sig i väsentlig grad, varför denna kompensering kan uppfylla våra krav.

6. OVERLAY, UPPSTARTNING ETC.

Programmet finns lagrat på dectape under namnet ROTSYN XCT och ROTSYN XXX. Filerna har byggts med ett länksystem som framgår av nedanstående figur.

LK1	LK2	LK3
INAB INPZ CHAB CHPZ LISTK LISTØ	ROTIO ROTAX ROTRIT INGAIN PLGAIN	STDATA MATRIX SAMPLA STEGSV
RESIDENT CODE		
ROTSYN, FB, POLMU, FAMU, GUIDE		

figur 6

FAMU och POLMU är två biblioteksrutiner, som emellertid inte finns lagrade i maskinen.

Bilaga 6.1 visar hur filerna byggts upp med CHAIN och ABS.

På samma band som XCT- och XXX-filerna finns också huvudprogrammet och subrutinerna (inkl. FAMU och POLMU) lagrade i både käll- och binärkod.

Uppstartning kan göras med batch-remsan EXECUTE-ROTSYN och med bandet på bandstation med nummer 1.

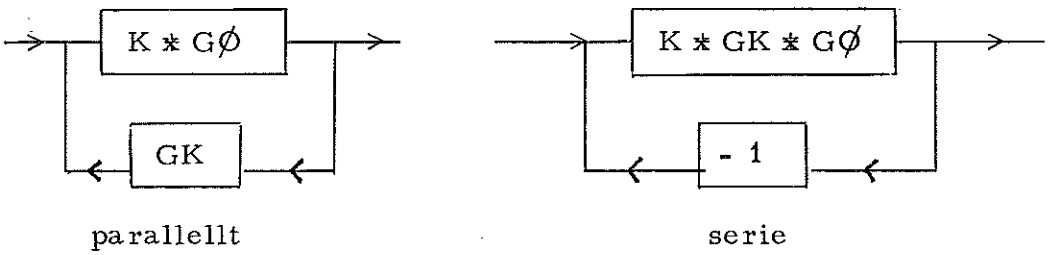
EXECUTE-ROTSYN har följande utseende:

```
$ JOB
PIP
T DK ← DT1 ROTSYN XCT (B)
T DK ← DT1 ROTSYN XXX (D)
$ JOB
E ROTSYN
$ EXIT
```

Har man ej tillgång till XCT- och XXX-filerna kan batch-remsan CAE-ROTSYN användas. Denna har ett utseende motsvarande bilaga 6.1 och kräver alltså tillgång på disken på huvudprogram och subrutiner i binärkod.

7. "USER'S GUIDE"

Programmet är avsett att användas som hjälpmedel vid syntes av linjära, tidsinvarianta system. Systemen antages bestå av två delar, dels det öppna systemet GØ, dels en kompenseringslänk GK. Kompenseringslänken kan införas i serie eller parallellt. Följande figur förtydligar.
K betecknar kretsförstärkningen



figur 7

Programmet finns lagrat på dectape som filerna ROTSYN XCT och ROTSYN XXX. Programmet kan startas upp med batch-remsan EXECUTE-ROTSYN och med bandet på bandstation med nummer 1. Finns ej tillgång till remsan måste man själv överföra filerna och därefter anropa EXECUTE.

I kommunikationen med användaren använder programmet följande beteckningar på GØ- och GK-variablerna. GØ och GK kan skrivas på två sätt, AB- och PZ-version.

$$G\emptyset(s) = K\emptyset \frac{s^{M\emptyset} + B\emptyset(1) * s^{M\emptyset-1} + \ldots + B\emptyset(M\emptyset)}{s^{N\emptyset} + A\emptyset(1) * s^{N\emptyset-1} + \ldots + A\emptyset(N\emptyset)} \qquad \text{(AB-version)}$$

eller

$$\ldots = K\emptyset \frac{(s-ZR\emptyset(1) - I * ZI\emptyset(1) * \ldots * (s-ZR\emptyset(M\emptyset) - I * ZI\emptyset(M\emptyset)}{(s-PR\emptyset(1) - I * PI\emptyset(1) * \ldots * (s-PR\emptyset(N\emptyset) - I * PI\emptyset(N\emptyset)} \quad \text{(PZ-version)}$$

GK skrivs helt analogt men med K i stället för Ø dvs. KK, MK, BK etc. i stället för KØ, MØ, BØ etc. Båda versionerna är helt likvärdiga och finns hela tiden tillgängliga även om enbart ena versionen matats in.

Högsta tillåtna gradtal är 1Ø, dvs. MØ, NØ, MK och NK måste alla vara mindre än eller lika med 1Ø. Samtidigt måste emellertid totala systemets överföringsfunktion ha högst gradtalet 1Ø i täljare och nämnare. Detta innebär att MØ+MK≤1Ø, NØ+NK≤ 1Ø samt vid parallellkompensering dessutom MØ+NK≤ 1Ø.

Användaren styr exekveringen genom att använda en uppsättning kommandon som skrives på teletypen. Kommandona tolkas av huvudprogrammet som anropar avsedd(a) rutin(er).

Programmet är berett att ta emot ett kommando när det gjort följande utskrift på teletypen:

YOUR COMMAND:

>

Vilka kommandon har då användaren tillgängliga och vilken effekt har dom? Nedan följer en redogörelse av dessa.

GUIDE -ger en utskrift som beskriver beteckningarna i GØ och GK enligt ovan. Vidare en förteckning över tillgängliga kommandon och några ord om deras innebörd. Utskriftsenheten kan vara teletype, display eller radskrivare. Vilken som ska väljas bestämmer användaren genom att svara på:

WRITE WANTED DEVICE: TT, LP OR VP

>

INAB -inputrutin för GØ och GK givna i AB-version. Inmatningen sker enligt följande. Först frågas efter om inmatningen avser GØ eller GK:

GØ OR GK

>

Därefter vill programmet veta värdena på KØ, MØ, NØ alt. KK, MK, NK, exempelvis:

WRITE KØ, MØ, NØ

#

Sedan återstår att mata koefficienterna i täljare och nämnare (om respektive gradtal ej är lika med Ø), exempelvis:

WRITE AØ(1), ..., AØ(N)

#

Exempel: $G\emptyset(s) = 2/s^2$ matas alltså in på följande sätt

GØ OR GK

> GØ

WRITE KØ, MØ, NØ

2 Ø 2

WRITE AØ(1), ..., AØ(N)

Ø Ø

INPZ -är PZ-versionens motsvarighet till INAB. Inmatningen sker identiskt till en början. Men i stället för koefficienter i täljare och nämnare ska nollställen och poler matas in GK=1Ø(s+1)/(s+1Ø) matas in på följande sätt:

GØ OR GK

> GK

WRITE KK, MK, NK

1Ø 1 1

WRITE ZRK(1), ZIK(1), ... ZRK(M), ZIK(M)

ONLY ONE ROOT/LINE AND COMPLEX-CONJUGATED ALLOWED

-1 Ø

WRITE PRK(1), PIK(1), ..., PRK(N), PIK(N)

- 1Ø Ø

Som framgår måste alltså nollställen och poler vara komplexkonjugerade. Detta på grund av att annars blir inte AB-versionen reell.

CHAB -är en rutin för att ändra värdet på någon variabel i AB-versionerna. Gradtalen kan dock inte ändras. Inmatningen sker i följande ordning: variabelnamn, eventuellt index och nytt värde. Radbyte, parenteser, likhetstecken o.dyl. kan inplaceras efter eget önskemål. Om vi vill ändra värdet på BK(2) till 1.72 kan detta exempelvis göras på följande sätt:

> CHAB BK(2) = 1.72

> CHAB BK

2

1.72
> CHAB
WRITE VARIABLE, INDEX (IF VECTOR) AND NEW VALVE
> BK 2 1.72
Berörda variabler i PZ-versionen ändras automatiskt.

CHPZ -är analog med CHAB men avser variablerna i PZ-versionerna. Ändringen sker på samma sätt. Den komplexkonjugerade variabeln (om den finns) ändras automatiskt till att förbli så. Också berörda delar i AB-versionen ändras.

LISTØ -listar aktuella värden på GØ-variablerna på teletypen. Både AB- och PZ-versionen listas.

LISTK -GK:s motsvarighet till LISTØ

ROTORT -initierar ett antal rutiner som plottar rotorten. Först frågar programmet efter minsta och största värde på förstärkningen, dvs. mellan vilka gränser denna ska varieras:
WRITE MIN AND MAX GAIN
#
Därefter frågar programmet om användaren vill ha utmärkt rötternas läge för några speciella värden på förstärkningen med värdet på förstärkningen angivet:
SPECIAL VALUES OF THE GAIN?
TERMINATE WITH EMPTY LINE
#
Högst tio värden kan matas in.
När inmatningen är klar plottas rotorten.
Om vi exempelvis vill att förstärkningen ska variera mellan Ø och 5 och dessutom vill att värdena 2, 2.5 och 3 på förstärkningen ska utmärkas skulle detta innebära följande inmatning.
WRITE MIN AND MAX GAIN
Ø 5
SPECIAL VALUES OF THE GAIN?
TERMINATE WITH EMPTY LINE
2 2.5 3
#

STEGSVAR -initierar de subrutiner som plottar stegsvaret. När detta kommando givits vill programmet veta vilket värde på förstärkningen som ska användas:

WRITE WANTED GAIN

#

Programmet måste också veta vilken typ av kompensering som är aktuell (om GK = 1 är typerna likvärdiga och detta frågas ej).

SER. OR PAR. COMP:

>

Det räcker att svara med bokstaven S respektive P.

Om vi exempelvis vill att stegsvaret ska plottas för förstärkningen 1.5 och GK är kopplad i serie får vi följande inmatning:

WRITE WANTED GAIN

1.5

SER. OR PAR. COMP.

> S

När inmatningen är klar beräknas och plottas stegsvaret.

END -avslutar exekveringen, återhopp till operativsystemet.

BILAGOR

```
YOUR COMMAND:
>GUIDE
WRITE WANTED OUTPUT DEVICE:TT,LP OR VP
>TT
GØ(S)=THE OPEN SYSTEM AND GK(S)=THE COMPENSATION LINK ARE GIVEN AS:

        S**M+B(1)*S**(M-1)+...+B(M)
G(S) = K*----------------------------      (AB-VERSION)
        S**N+A(1)*S**(N-1)+...+A(N)

OR

        (S-ZR(1)-I*ZI(1))*...*(S-ZR(M)-I*ZI(M))
     = K*---------------------------------------      (PZ-VERSION)
        (S-PR(1)-I*PI(1))*...*(S-PR(N)-I*PI(N))

WHERE K,A,B,... =KØ,AØ,BØ,... IN GØ
                =KK,AK,BK,... IN GK

AVAILABLE COMMANDS:

GUIDE       -MAKES THIS OUTPUT

INAB,INPZ   -INPUT ROUTINES FOR GØ AND GK IN AB- RESP. PZ-VERSION.

CHAB,CHPZ   -ROUTINES TO CHANGE THE VALUE OF A VARIABLE IN GØ OR GK
              IN AB- RESP. PZ-VERSION. MØ,NØ,MK,NK CAN'T BE CHANGED.

LISTØ,LISTK-LISTS ON TT ALL GØ RESP. GK VARIABLES.

ROTORT      -INITIALIZES THE ROUTINES THAT PLOT THE ROOT LOCUS FOR
              THE SYSTEM GØ COMPENSATED WITH GK.

STEGSVAR    -INITIALIZES THE ROUTINES THAT PLOT THE STEGSVAR.

END         -TERMINATES EXECUTION.

YOUR COMMAND:
>
```

```
C       PROGRAM NAME:ROTSYN
C
C       PROGRAM FOR SYNTHESIS OF SYSTEMS,LINEAR AND INVARIABLE
C       IN TIME WITH THE ROOT-LOCUS METHOD.
C
C       THE MAIN PROGRAM READS USER'S COMMANDS FROM TT AND
C       CALLS SUBROUTINES FOR COMPUTATION.
C
C       THE SUBROUTINES CAN BE SEPARATED IN THE FOLLOWING GROUPS:
C       1.EXPLAIN VARIABLES USED AND COMMANDS AVAILABLE (GUIDE).
C       2.INPUT ROUTINES FOR THE OPEN SYSTEM AND THE CONTROLLER
C         (INAB,INPZ).
C       3.CHANGE THE VALUE OF A VARIABLE (CHAB,CHPZ).
C       4.LIST VALUES OF THE VARIABLES (LISTO,LISTK).
C       5.PLOT THE ROOT LOCUS OF THE SYSTEM (FB,ROTIO,ROTAX,
C         ROTRIT,INGAIN,PLGAIN).
C       6.PLOT THE STEGSVAR OF THE SYSTEM (FB,STDATA,MATRIX,
C         SAMPLA,STEGSV).
C
C       AUTHOR:BO KENNEDY 1973-04-07
C
C       SUBROUTINES REQUIRED:
C          GUIDE,INAB,INPZ,CHAB,CHPZ,LISTO,LISTK,FB,ROTIO,ROTAX,ROTRIT,
C          INGAIN,PLGAIN,STDATA,MATRIX,SAMPLA,STEGSV,
C          ROT,FAMU,POLMU,COSA,
C          SCALE,AXIS,PLOTTA,MARK,NUMBER,LINE,ERASE,
C          RLINE,FAC,RFP,RABC,RTTFF,IRTTFF
      REAL K0,KK
      DIMENSION C0(10)
      COMMON /OPEN/ K0,M0,N0,A0(10),B0(10),ZR0(10),ZI0(10),
     *PR0(10),PI0(10)
      COMMON /COMP/ KK,MK,NK,AK(10),BK(10),ZRK(10),ZIK(10),
     *PRK(10),PIK(10)
      COMMON /TAELJ/ M,E(21)
      COMMON /NAEMN/ N,C(21),D(21)
      COMMON /ICHAB/ I,BUFF(16)
      COMMON /MATR/ A(10,10),B(10,1),TS,TF
      COMMON /AXDATA/ DS,XSMIN,YSMIN,SKMIN,SKMAX
      COMMON /STDAT/ IFB,SK
      COMMON /GAINS/ SL(10),IANTAL
```

```
      SUBROUTINE GUIDE
C     IF WANTED THIS ROUTINE EXPLAINS THE COEFFICIENTS USED
C     IN THE COMMUNICATION BETWEEN THE USER AND THE PROGRAM
C     A BRIEF DESCRIPTION OF THE COMMANDS AVAILABLE IS ALSO GIVEN
C     OUTPUT IS MADE ON TT,LP OR VP WHATEVER WANTED
C
C     SUBRUT. REQ.: RLINE,RABC
      COMMON /SLASK/ BUFF(16),I,HOLL,IND
```

```
      SUBROUTINE INAB
C     INPUT ROUTINE FOR G0 AND GK VARIABLES IN AB-VERSION.
C     FIRST THE ROUTINE ASKS ON TT IF G0 OR GK INPUT
C     EG.       G0 OR GK
C               >GK
C     AFTER THAT IT WANTS VALUES ON KK,MK,NK OR K0,M0,N0
C     DEPENDING ON THE PREVIOUS ANSWER
C     EG.       WRITE KK,MK,NK
C               □1 0 2
C     THEN IT WANTS VALUES ON AK,BK OR A0,B0 VECTORS (IF
C     NK,MK,N0,M0 RESP. .NE.0)
C     EG.       WRITE AK(1),...,AK(M)
C               □0 0
C
C     THIS EXAMPLE GIVES GK=1/S**2
C
C     INPUT IS MADE IN FREE FORMAT
C     MAX VALUE OF M0,N0,MK,NK IS 10
C
C     SUBRUT. REQ.:RLINE,FAC,RTTFF,IRITFF,ROT
      REAL K,K0,KK
      COMMON /SLASK/ A(10),B(10),BUFF(16),I,CHAR,ITYPE,IC,K,M,N,IT,EPS
      COMMON /OPEN/ K0,M0,N0,A0(10),B0(10),ZR0(10),ZI0(10),
     *PR0(10),PI0(10)
      COMMON /COMP/ KK,MK,NK,AK(10),BK(10),ZRK(10),ZIK(10),
     *PRK(10),PIK(10)



      SUBROUTINE INPZ
C     INPUT ROUTINE FOR G0 AND GK VARIABLES IN PZ-VERSION
C     THE G0/GK AND KK,MK,NK/K0,M0,N0 INPUTS ARE IDENTICAL
C     TO THOSE IN SUBRUT. INAB
C     AFTER THESE INPUTS THE ROUTINE ASKS FOR THE ZEROES
C     AND POLES FOR G0 RESP. GK. THESE MUST BE WRITTEN WITH
C     ONE ZERO OR POLE PER LINE, BOTH REAL AND IMAGINAR PARTS
C     MUST BE WRITTEN EVEN IF .EQ.0
C     ONLY COMPLEX-CONJUGATED ZEROES AND POLES ARE ALLOWED.
C     EG.       G0 OR GK
C               >G0
C               WRITE K0,M0,N0
C               □1 1 2
C               WRITE ZR0(1),ZI0(1),...,ZR0(M),ZI0(M)
C               ONLY ONE ROOT/LINE AND COMPLEX-CONJUGATED ALLOWED
C               □-0.5 0
C               WRITE PR0(1),PI0(1),...,PR0(N),PI0(N)
C               ONLY ONE ROOT/LINE AND COMPLEX-CONJUGATED ALLOWED
C               □-2 1
C               □-2 -1
C
C     THIS EXAMPLE GIVES G0=(S+0.5)/((S+2+I)*(S+2-I))
C
C     INPUT IS MADE IN FREE FORMAT
C     MAX VALUE OF M0,N0,MK,NK IS 10
C
C     SUBRUT. REQ.:RLINE,FAC,RTTFF,IRTTFF,FAMU
      REAL K,K0,KK
      COMMON /SLASK/ A(10),B(10),C(10),BUFF(16),I,CHAR,ITYPE,IC,K,M,N,EPS
      COMMON /OPEN/ K0,M0,N0,A0(10),B0(10),ZR0(10),ZI0(10),
     *PR0(10),PI0(10)
      COMMON /COMP/ KK,MK,NK,AK(10),BK(10),ZRK(10),ZIK(10),
     *PRK(10),PIK(10)
```

```
      SUBROUTINE CHAB
C     ROUTINE TO CHANGE THE VALUE OF A VARIABLE IN G0 OR GK
C     IN AB-VERSION
C     NB. M0,N0,MK,NK CAN'T BE CHANGED BY THIS ROUTINE
C     THE PZ-VERSION VARIABLES CONCERNED ARE AUTOMATICALLY CHANGED
C     INPUT CAN BE MADE IN DIFFERENT WAYS IF ONLY THE FIRST INPUT
C     IS VARIABLE, SECOND INDEX (IF VECTOR) AND LAST NEW VALUE
C     EG.       >CHAB
C               WRITE VARIABLE,INDEX (IF VECTOR) AND NEW VALUE
C               >A0 2 3.2
C     EG.       >CHAB A0(2)=3.2
C     EG.       >CHAB A0
C               ?2 3.2
C
C     (,),= CAN BE WRITTEN OR LEFT OUT WHATEVER WANTED
C     INPUT IS MADE IN FREE FORMAT
C
C     SUBRUT. REQ.: RLINE,FAC,RABC,RFP,ROT
      REAL K0,KK
      DIMENSION S(4),R(4)
      COMMON /SLASK/ IT,EPS,HOLL,IND,CHAR1,ITYPE1,CHAR2,ITYPE2,
     *J,FRES,IRES,INDEX
      COMMON /OPEN/ K0,M0,N0,A0(10),B0(10),ZR0(10),ZI0(10),
     *PR0(10),PI0(10)
      COMMON /COMP/ KK,MK,NK,AK(10),BK(10),ZRK(10),ZIK(10),
     *PRK(10),PIK(10)
      COMMON /ICHAB/ I,BUFF(16)



      SUBROUTINE CHPZ
C     ROUTINE TO CHANGE THE VALUE OF A VARIABLE IN G0 OR GK
C     IN PZ-VERSION
C     NB. M0,N0,MK,NK CAN'T BE CHANGED BY THIS ROUTINE
C     THE AB-VERSION VARIABLES AND COMPLEX-CONJUGATED (IF EXISTING)
C     ZEROES AND POLES ARE AUTOMATICALLY CHANGED
C     INPUT CAN BE MADE IN MANY WAYS IF ONLY THE FIRST INPUT IS
C     VARIABLE, SECOND INDEX (IF VECTOR) AND LAST NEW VALUE
C     EG.    >CHPZ
C            WRITE VARIABLE,INDEX (IF VECTOR) AND NEW VALUE
C            THE COMPLEX-CONJUGATED ROOT (IF EXISTING) IS AUTOMATICALLY CHANGED
C            PRK(2)=-5.7
C     EG.    >CHPZ K0 1.2
C     EG.    >CHPZ PI0
C            ?3 2.3
C
C     (,),= CAN BE WRITIEN OR LEFT OUT WHATEVER WANTED
C     INPUT IS MADE IN FREE FORMAT
C
C     SUBRUT. REQ.: RLINE,RABC,RFP,FAMU
      REAL K0,KK
      DIMENSION R(4),S(10)
      COMMON /OPEN/ K0,M0,N0,A0(10),B0(10),ZR0(10),ZI0(10),
     *PR0(10),PI0(10)
      COMMON /COMP/ KK,MK,NK,AK(10),BK(10),ZRK(10),ZIK(10),
     *PRK(10),PIK(10)
      COMMON /ICHAB/ I,BUFF(16)
      COMMON /SLASK/ EPS,HOLL,IND,J,FRES1,IRES1,FRES2,IRES2,X(10)
```

```
      SUBROUTINE LISTO
C     LISTS ON TT THE VALUES OF THE GO-VARIABLES
C     SUBRUT. REQ.: NONE
      REAL KO
      COMMON /OPEN/ KO,MO,NO,AO(10),BO(10),ZRO(10),ZIO(10),
     *PRO(10),PIO(10)
```

```
      SUBROUTINE LISTK
C     LISTS ON TT THE VALUES OF THE GK-VARIABLES
C     SUBRUT. REQ.: NONE
      REAL KK
      COMMON /COMP/ KK,MK,NK,AK(10),BK(10),ZRK(10),ZIK(10),
     *PRK(10),PIK(10)
```

```
      SUBROUTINE FB
C     ROUTINE TO COMPUTE G(S) FOR THE COMPENSATED SYSTEM, IE. MULTIPLY
C     NUMERATORS AND DENOMINATORS IN GO(S) AND GK(S)IN COMBINATIONS
C     DETERMINED BY THE TYPE OF COMPENSATION. THE INPUT VARIABLE IT
C     CONTROLS COMPUTATION IN THIS RESPECT:
C     IT=1 SERIAL COMPENSATION
C        2 PARALLELL    "
C        3 ONLY DENOMINATOR COMPUTED
C     THE ROUTINE GIVES G(S) IN THE FORM:
C                  E(1)*S**M+...+E(M+1)
C     G(S)=----------------------------------------
C          (C(1)+K*D(1))*S**N+...+(C(N+1)+K*D(N+1))
C
C     SUBRUT. REQ.:POLMU,IRTTFF
      REAL K,KO,KK
      DIMENSION A(11),B(11)
      COMMON /OPEN/ KO,MO,NO,AO(10),BO(10),ZRO(10),ZIO(10),P
     *RO(10),PIO(10)
      COMMON /COMP/ KK,MK,NK,AK(10),BK(10),ZRK(10),ZIK(10),P
     *RK(10),PIK(10)
      COMMON /TAELJ/ M,E(21)
      COMMON /NAEMN/ N,C(21),D(21)
      COMMON /SIDAT/ IT,SK
```

```
      SUBROUTINE ROTIO
C     INPUT ROUTINE FOR THE MIN AND MAX VALUES OF THE GAIN FOR WHICH
C     THE ROOT LOCUS IS WANTED TO BE PLOTTED
C     FROM THESE VALUES THE ROUTINE CALCULATES THE VALUES
C     NECESSARY TO DRAW THE AXIS FOR THE ROOT LOCUS
C     EG.       WRITE MIN AND MAX GAIN
C               DO 7
C
C     SUBRUT. REQ.:RTTFF,ROT,SCALE
      COMMON /NAEMN/ N,C(21),D(21)
      COMMON /AXDATA/ DS,XSMIN,YSMIN,SKMIN,SKMAX
      COMMON /SLASK/ AR(11),ZR1(10),ZI1(10),BUFF(16),IR(10)
     *,EPS,IC,X,I,IT,J,NPOI,UMAX,UMIN,XDS,YDS

      SUBROUTINE ROTAX
C     DRAWS THE AXIS FOR THE ROOT LOCUS
C     NECESSARY DATA ARE GIVEN BY THE ROUTINE
C     ROTIO THROUGH COMMON /AXDATA/
C
C     SUBRUT. REQ.: AXIS,MARK
      COMMON /SLASK/ IX,IY,I,J
      COMMON /AXDATA/ DS,XSMIN,YSMIN,SKMIN,SKMAX

      SUBROUTINE ROTRIT
C     ROUTINE TO PLOT THE ROOT LOCUS FOR THE SYSTEM G0 COMPENSATED
C     WITH GK.
C     NECESSARY DATA (SCALE FACTORS,VALUES OF THE GAIN ETC) ARE
C     PROVIDED BY THE ROUTINE ROTIO THROUGH COMMON /AXDATA/
C     AXIS ARE DRAWN BY THE ROUTINE ROTAX
C     PRINCIPAL:PLOT THE ROOTS FOR MIN GAIN. CHOSE A NEW VALUE
C     OF THE GAIN AND COMPUTE THE DISTANCES THE ROOTS HAVE MOVED
C     ADJUST THE CHANGE OF GAIN UNTIL THE ROOTS HAVE MOVED A
C     SUITABLE DISTANCE. PLOT THE ROOTS AND REPEAT PROCEDURE
C     UNTIL MAX VALUE OF THE GAIN IS ACHIEVED.
C
C     SUBRUT. REQ.: ROT,LINE,MARK
      REAL MAXMIN
      COMMON /SLASK/ AR(100),ZR(10),ZI(10),IR(10),II(10),MAXMIN,
     *IR1(100),II1(100),ICNTRL,EPS,IT,DIST,SK,X,I,SK1,K
      COMMON /NAEMN/ N,C(21),D(21)
      COMMON /AXDATA/ DS,XSMIN,YSMIN,SKMIN,SKMAX

      SUBROUTINE INGAIN
C     INPUT ROUTINE FOR VALUES OF THE GAIN FOR WHICH THE ROUTINE
C     PLGAIN PLOTS THE ROOTS. MAX 10 INPUT VALUES ALLOWED.
C
C     SUBRUT. REQ.:RLINE,RFP
      COMMON /SLASK/ BUFF(16),ICTRL,I,J,IRES,IND
      COMMON /GAINS/ SK(10),IANTAL

      SUBROUTINE PLGAIN
C     PLOTS THE ROOTS FOR THE GAINS INPUT IN THE ROUTINE INGAIN.
C     THE PLOTTING IS MADE IN AXIS GIVEN BY THE ROUTINE ROTAX.
C
C     SUBRUT. REQ.:ROT,LINE,MARK,NUMBER
      COMMON /SLASK/ AR(10),ZR(10),ZI(10),IR(10),II(10),
     *IT,EPS,I,J,X,K
      COMMON /NAEMN/ N,C(21),D(21)
      COMMON /AXDATA/ DS,XSMIN,YSMIN,SKMIN,SKMAX
      COMMON /GAINS/ SK(10),IANTAL
```

```
      SUBROUTINE STDATA
C     IN THIS ROUTINE THE USER GIVES VALUES TO PARAMETERS
C     USED WHEN PLOTTING THE STEGSVAR
C     IFB=1    -SERIAL COMPENSATION
C         2    -PARALLELL    "
C     SK       -WANTED GAIN
C
C     SUBRUT. REQ.: RTTFF,RLINE,RABC
      REAL KK
      COMMON /COMP/ KK,MK,NK,AK(10),BK(10),ZRK(10),ZIK(10),
     *PRK(10),PIK(10)
      COMMON /SLASK/ BUFF(16),IC,HOLL,IND
      COMMON /STDAT/ IFB,SK



      SUBROUTINE MATRIX
C     COMPUTES THE SAMPLING RATE TS AND THE TIME TO FOLLOW Y(T)
C     TF=100.0*TS ACCORDING TO THESE PRINCIPS:
C     Y(T) IS FOLLOWED FIVE PERIODS OR UNTIL ABS(Y(T)-1.0),LT.0.1%,
C     IF Y(T) IS NON-OSC. AND UNSTABLE Y(T) IS FOLLOWED UNTIL Y(T)=1.5
C     COMPUTES THE CONTINOUS MATRIXES A AND B FROM G(S)
C
C     SUBRUT. REQ.:ROT
      COMMON /SLASK/ ZR(20),ZI(20),H(20),EPS,PI,P,X,I,ZRMAX,ZIMAX,
     *INDEX,J
      COMMON /TAELJ/ M,E(21)
      COMMON /NAEMN/ N,C(21),D(21)
      COMMON /MATR/ A(10,10),B(10,1),TS,TF
      COMMON /STDAT/ IFB,SK



      SUBROUTINE SAMPLA
C     COMPUTES THE SAMPLED MATRIXES A AND B FROM THE CONTINOUS ONES
C
C     SUBRUT. REQ.:COSA
      COMMON /NAEMN/ N,C(21),D(21)
      COMMON /MATR/ A(10,10),B(10,1),TS,TF



      SUBROUTINE STEGSV
C     COMPUTES THE STEGSVAR BY ITERATING THE SAMPLED STATE
C     EQUATIONS WHERE
C     X-VECTOR     -IS THE STATE AT THE TIME T
C     X1-VECTOR    -IS THE STATE AT THE TIME T+TS
C     THE OUTPUT Y(T)=X(1) IS PLOTTED VERSUS THE TIME T
C
C     SUBRUT. REQ.:AXIS,SCALE,MARK,PLOTTA
      COMMON /SLASK/ X(10),DS,IFNT,SMIN,IX(2),I,K,J,X1(10),IY(1)
      COMMON /MATR/ A(10,10),B(10,1),TS,TF
      COMMON /NAEMN/ N,C(21),D(21)
```

```
$E ROTSYN


THE COMMAND GUIDE WILL GIVE YOU AN EXPLAINATION
OF THE COEFFICIENTS USED
A BRIEF DESCRIPTION OF THE COMMANDS AVAILABLE IS ALSO GIVEN

YOUR COMMAND:
>
INAB
GØ OR GK
>GØ
WRITE KØ,MØ,NØ
#1 Ø 2
WRITE AØ(1),...,AØ(N)
#Ø Ø

YOUR COMMAND:
>INAB
GØ OR GK
>GK
WRITE KK,MK,NK
#1 Ø Ø

YOUR COMMAND:
>ROTORT
WRITE MIN AND MAX GAIN
#Ø 5
SPECIAL VALUES OF THE GAIN?
TERMINATE WITH EMPTY LINE
#2 4
#

YOUR COMMAND:
>STEGSV
WRITE WANTED GAIN
#1

YOUR COMMAND:
>INAB
GØ OR GK
>GK
WRITE KK,MK,NK
#1 1 Ø
WRITE BK(1),...,BK(M)
#1

YOUR COMMAND:
>ROTORT
WRITE MIN AND MAX GAIN
#Ø 5
SPECIAL VALUES OF THE GAIN?
TERMINATE WITH EMPTY LINE
#2 4
#

YOUR COMMAND:
>
```

```
 STEGSV
WRITE WANTED GAIN
#2
SER. OR PAR. COMP.
>S

YOUR COMMAND:
>INAB
GØ OR GK
>GK
WRITE KK,MK,NK
#1Ø 1 1
WRITE BK(1),...,BK(M),AK(1),...,AK(N)
#1 1Ø

YOUR COMMAND:
>ROTORT
WRITE MIN AND MAX GAIN
#Ø 4
SPECIAL VALUES OF THE GAIN?
TERMINATE WITH EMPTY LINE
#2 3 3.5
#

YOUR COMMAND:
>STEGSV
WRITE WANTED GAIN
#2
SER. OR PAR. COMP.
>S

YOUR COMMAND:
>END

STOP  ØØØØØØ


DOS-15 V1A
$
```

RE(Z)
3.75E+0
2.5E+0
1.25E+0
0.0E+0
-1.25E+0
-2.5E+0
IM(Z)
-2.5E+0
-1.25E+0
0.0E+0
1.25E+0
1.
2.

1
5.0E+1
4.0E+1
3.0E+1
2.0E+1
1.0E+1
0.0E+0
2.00E-1
6.00E-1
1.00E+0
1.40E+0
Y(I)

RE(Z)
1.0E+0
0.0E+0
-1.0E+0
-2.0E+0
-3.0E+0
-4.0E+0
IM(Z)
1.0E+0
0.0E+0
-1.0E+0
-2.0E+0
2.
1.

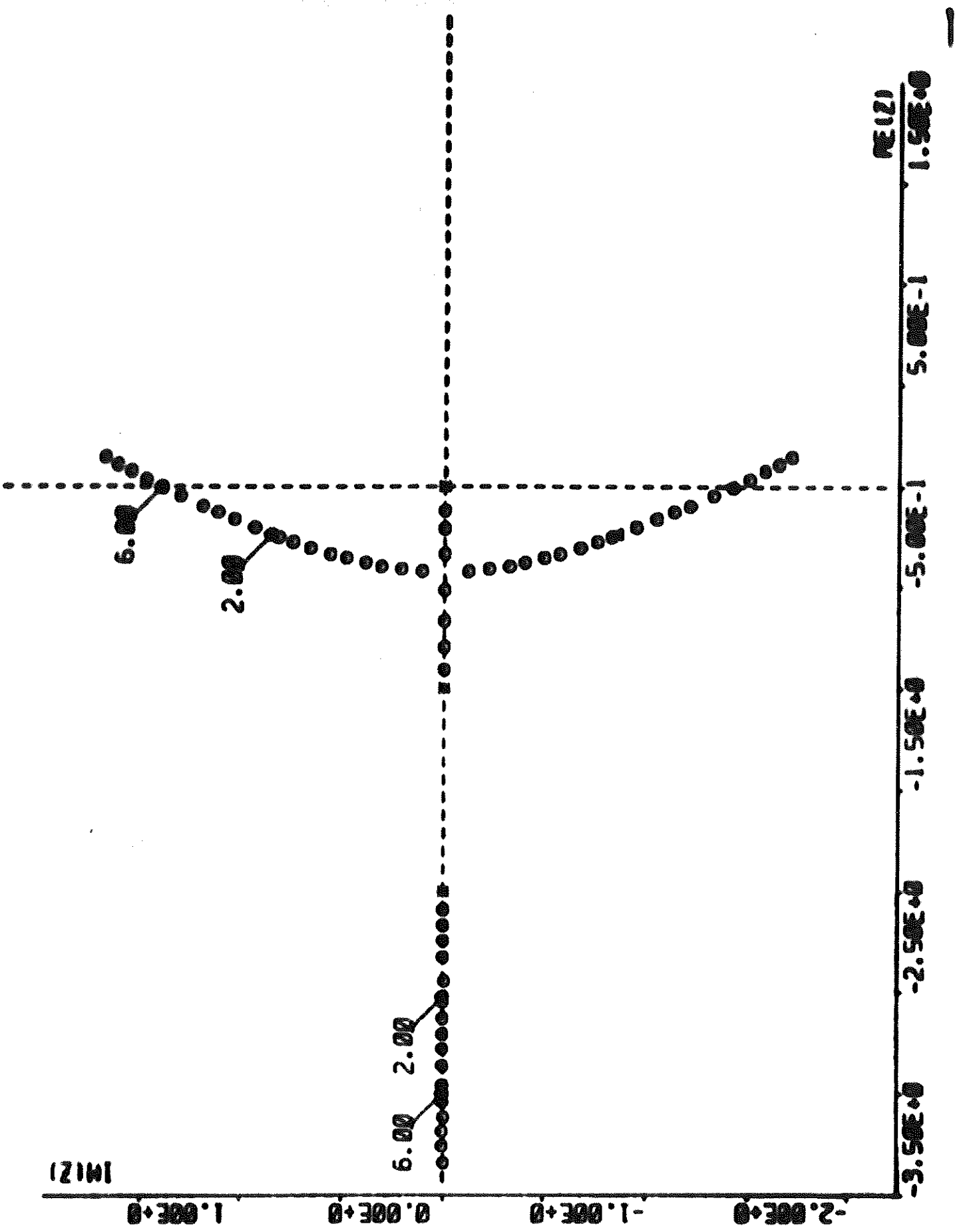
1.0E+1
8.0E+0
6.0E+0
4.0E+0
2.0E+0
0.0E+0
2.00E-1
6.00E-1
1.00E+0
1.40E+0

2.
3.
3.5
3.5
3.
3.
2.00
IM(Z)
RE(Z)
0.
-2.
-4.
-6.
-8.
-1.
-4.00E+0
-2.00E+0
0.00E+0
2.00E+0

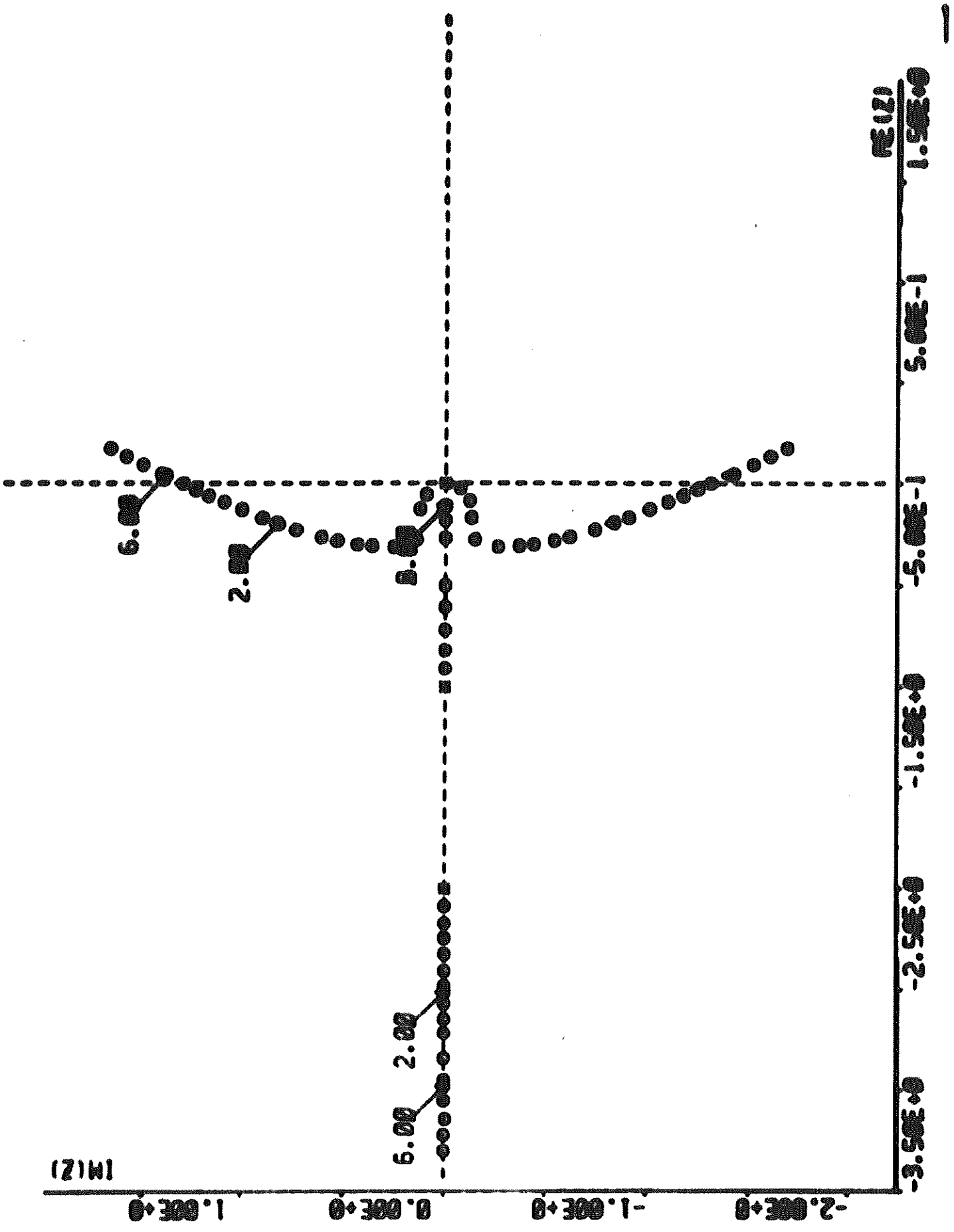
1.0E+1
8.0E+0
6.0E+0
4.0E+0
2.0E+0
0.0E+0
2.0E-1
6.0E-1
1.0E+0
1.4E+0
V(T)

```
$E ROTSYN


THE COMMAND GUIDE WILL GIVE YOU AN EXPLAINATION
OF THE COEFFICIENTS USED
A BRIEF DESCRIPTION OF THE COMMANDS AVAILABLE IS ALSO GIVEN

YOUR COMMAND:
>
INAB
GØ OR GK
>GØ
WRITE KØ,MØ,NØ
#1 Ø 3
WRITE AØ(1),...,AØ(N)
#3 2 Ø

YOUR COMMAND:
>INAB
GØ OR GK
>GK
WRITE KK,MK,NK
#1 Ø Ø

YOUR COMMAND:
>ROTORT
WRITE MIN AND MAX GAIN
#Ø 1Ø
SPECIAL VALUES OF THE GAIN?
TERMINATE WITH EMPTY LINE
#2 6
#

YOUR COMMAND:
>STEGSV
WRITE WANTED GAIN
#2

YOUR COMMAND:
>INAB
GØ OR GK
>GK
WRITE KK,MK,NK
#1 1 1
WRITE BK(1),...,BK(M),AK(1),...,AK(N)
#Ø.1 Ø

YOUR COMMAND:
>ROTORT
WRITE MIN AND MAX GAIN
#Ø 1Ø
SPECIAL VALUES OF THE GAIN?
TERMINATE WITH EMPTY LINE
#2 6
#

YOUR COMMAND:
>
```

```
 STEGSV
WRITE WANTED GAIN
#2
SER. OR PAR. COMP.
>S

YOUR COMMAND:
>STEGSV
WRITE WANTED GAIN
#1
SER. OR PAR. COMP.
>S

YOUR COMMAND:
>END

STOP  000000


DOS-15 V1A
$
```


IM(Z)
RE(Z)
6.00
2.00
2.00
6.00

5.12


IM(Z)
RE(Z)
6.
2.
1.
2.00
6.00

2.00E-1
6.00E-1
1.00E+0
1.40E+0
Y(I)
0.00E+0
1.00E+1
2.00E+1
3.00E+1
4.00E+1
5.00E+1
I

1
5.0E+1
4.0E+1
3.0E+1
2.0E+1
1.0E+1
0.0E+0
2.0E-1
6.0E-1
1.0E+0
1.4E+0

```
$B PR

DOS-15 V1A
$$JOBCHAIN

CHAIN LP1

NAME XCT FILE
>ROTSYN
LIST OPTIONS & PARAMETERS
>NM
DEFINE RESIDENT CODE
>ROTSYN,FB,POLMU,FAMU,GUIDE
DESCRIBE LINKS & STRUCTURE
>LK1=INAB,INPZ,CHAB,CHPZ,LISTØ,LISTK
>LK2=ROTIO,ROTAX,ROTRIT,INGAIN,PLGAIN
>LK3= STDATA,MATRIX,SAMPLA,STEGSV
>LK1:LK2:LK3
>

      CORE REQ'D
        4Ø723-77636

DOS-15 V1A
$$JOBABS

ABSOLUTE V1A
ENTER FILE NAME  >ROTSYN
DOS-15 V1A
$$JOBE ROTSYN
```