BESTÄMNING AV ÖVERFÖRINGSFUNKTION MED IMPULSSVARSANALYS

EN TILLÄMPNING AV PARAMETRISK OPTIMERING

Examensarbete vid Institutionen för Regleringsteknik
Tekniska Högskolan i Lund, läsåret 1968-69.

Bertil Lundgren

INNEHÅLLSFÖRTECKNING

CALCULATION OF TRANSFER-FUNCTION WITH PULSERESPONSEANALYSIS

The problem above has been studied at the Division of Automatic Control, Lund Institute of Technology.

PROBLEM

Assume we have a pulseresponse from a system. We measure the pulse-response in discrete points. The value in point number k is called YM_k. Then we estimate the order and the parameters of the system. We calculate the pulse-response from theese estimated parameters, and the calculated corresponding to YM_k is called YC_k. Now we derive a lossfunction $V=\sum_{k=1}^{M}(YC_k-YM_k)^2$, where M is the number of measurepoints. V is a function of the parameters of the system. If the order and the parameters of the system were correct, V should be very small. How small depends on the error in YM_k. By minimizing V we can get the correct parameters. If V can not be small enough, the order of the estimated system is to small.

Two iterative minimization-algorithms has been used, Newton-Raphsons method (NR) and Fletcher-Powells method (FP). Both methods use the gradient of the function and the inverse of the second-derivative-matrix in every iteration. In (NR) the second-derivative-matrix G is calculated in the ordinary manner and then G is inverted. In (FP) the corresponding to G^{-1} is calculated with an iterative method. No inversion is made.

RESULTS

Two pulse-responses has been studied. Both systems were identified with good accuracy by a fifth order system. The identifications were made in 3 minutes at the Uppsala-computer CD 3600, with addition-time 4.25 microseconds.

Comparation of the minimization-methods: (FP) is not more rapid than (NR) although no inversion of the matrix is made. In both methods a linear minimization is involved. If (FP) shall converge, the linear minimization must be done very accurate. In (NR) the accuracy of the linear minimization is not important.

Bertil Lundgren

REFERENSER

REF 1 Johnsson-Rosell
Examensarbete: Identifiering av lineära system med hjälp av impulssvar

REF 2 Computer Journal vol 6, 1963

REF 3 Walsh
Introduction to numerical analysis

REF 4 SIAM Review vol 4 sid 343

REF 5 K J Åström
On the achievable accuracy in identification-problem

KAP 1. INLEDNING

1. Problemställning

Målet för examensarbetet är att utarbeta en metod för att bestämma överföringsfunktionen hos ett linjärt, tidsinvariant system. Man kan också säga att man vill kunna identifiera ett system med de nämnda egenskaperna. Ett FORTRAN-program, som löser problemet praktiskt, skall också skrivas.

Man kan gå till väga på olika sätt. I det här fallet vill man kunna beräkna överföringsfunktionen ur systemets impulssvar. Impulssvaret betecknas YM, dvs den uppmätta utsignalen.

Identifieringen kan ske på följande sätt:

Man uppskattar ordningstalet N och parametrarna hos systemet, och beräknar ett impulssvar som betecknas YB. Impulssvaren avläses vid diskreta tidpunkter, och man kan ställa upp följande förlustfunktion

$$V = \sum_{k=1}^{M} (YB_k - YM_k)^2$$

där YB_k och YM_k är respektive impulssvars värde vid mätpunkt nummer k, och M är antalet mätpunkter. Eftersom YB_k är en funktion av systemparametrana, blir V det också. OM man använt de korrekta parametervärdena vid beräkningen av YB_k , så blir V litet i någon mening. Man kan alltså identifiera systemet genom att minimera funktionen V. Om inte V kan fås tillräckligt litet, försöker man med ett högre ordningstal.

Om det finns störningar på utsignalen, så kan inte V fås exakt lika noll. Man kan då avgöra om man identifierat systemet med ett statistiskt test enligt REF 5. Jag har inte undersökt den här frågan, eftersom jag hela tiden använt deterministiska impulssvar.

Problemet att identifiera ett system har nu överförts till problemet att minimera en funktion, som kommer att ha 2N oberoende variabler.

I examensarbetet har använts två iterativa minimeringsmetoder:

Newton-Raphsons metod
Fletcher-Powells metod

Båda metoderna använder sig av funktionens gradient vid varje iteration. I Newton-Raphsons metod beräknas även andraderivatmatrisen och dess invers i varje iteration. I Fletcher-Powells metod fås motsvarigheten till andraderivatmatrisens invers fram på iterativ väg. Se vidare kap 2.

Ett liknande problem har undersökts av Johnsson-Rosell i deras examensarbete, REF 1.

2. Examensarbetets omfattning

Jag har skrivit ett FORTRAN-program, som utför identifieringsproblemet. En subrutin, som plottar de båda utsignalerna på radskrivaren ingår också i programmet.

I båda minimeringsalgoritmerna ingår också en linjär minimering. Tre olika varianter av den linjära minimeringen har jämförts. Minimeringsrutinerna går utmärkt att använda separat vid minimering av mera "normala funktioner" av flera variabler, t ex polynom. Se kap 6.

3. Sammanfattning av resultat

I examensarbetet har inte utförts några praktiska mätningar, utan mätresultaten härför från institutionen för Teletransmissionsteori.

Överföringsfunktionen till två system har beräknats. Impulssvarens utseende framgår av fig 1.1 och fig 1.2.

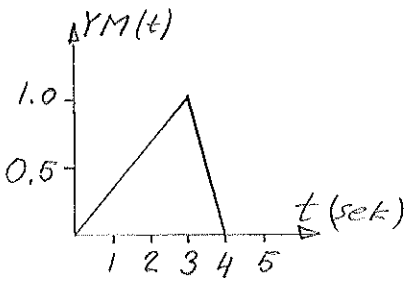


Fig 1.1 Impulssvar 1

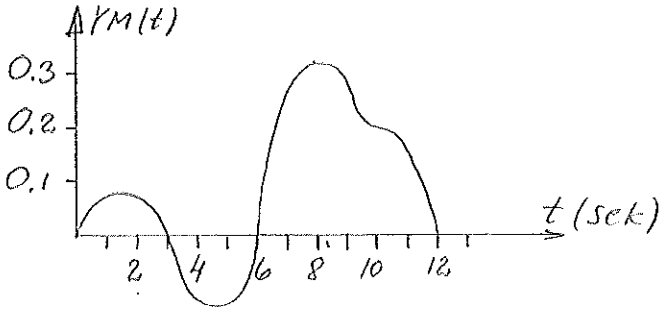


Fig 1.2 Impulssvar 2

Ett femte ordningens system identifierade godtagbart Impulssvar 1. Förlustfunktionen blev 0.069 , då 40 mätpunkter användes. Se sid 6:5.

För Impulssvar 2 gav ett femte ordningens system förlustfunktionen 0.028 med 56 mätpunkter. Se sid 6:11.

Båda identifieringarna utfördes på 3 min på Uppsalamaskinen CD 3600. Maskinens räknetid för flytande addition är 4.25 mikrosek.

Beträffande de två minimeringsmetoderna har framkommit följande:

Newton-Raphsons metod kräver ingen noggrann linjär minimering. SUBROUTINE LINMINB duger utmärkt. Om andraderivatmatrisen skulle bli singulär, måsta man göra något åt saken. Detta finns inlagt i programmet, varför den nackdelen hos Newton-Raphsons metod är eliminerad.

Med Fletcher-Powells metod är den linjära minimeringen av avgörande betydelse för metodens konvergens. Metoden konvergerar inte med en dålig linjär minimering. SUBROUTINE LINMINA eller LINMINC måste användas.

Vilken metod räknar då snabbast? Vid en noggrann linjär minimering måste förlustfunktionen beräknas ett antal gånger. Att beräkna förlustfunktionen för ett dynamiskt system är en komplicerad procedur. Kanske tiden man tjänar på att slippa invertera en matris ätes upp av tiden för den noggrannare minimeringen?

Så är troligen fallet. Räknetiden har inte vid någon identifiering blivit mindre med Fletcher-Powells metod. Räknetiderna är tämligen likartade.

KAP 2. NEWTON-RAPHSON FLETCHER-POWELL

TVÅ MINIMERINGSMETODER

1. Inledning

Det har utarbetats många metoder för att beräkna ett (lokalt) minimum till en funktion av flera variabler. I detta examensarbetet skall vi använda och jämföra två metoder:

NEWTON-RAPHSONS generaliserade metod

FLETCHER-POWELLS metod

Newton-Raphsons generaliserade metod är en utveckling av den gamla beprövade metoden för endimensionell minimering.

Fletcher-Powells metod presenterades år 1963 i REF 2, Computer Journal vol. 6.

I programmet återfinns de båda minimeringsalgoritmerna i var sin subrutin, nämligen NEWRAPS resp. FLEPO.

2. Härledning av allmän minimeringsalgoritm

Först skall vi diskutera minimeringsproblemet allmänt och härleda en iterationsformel, som gäller för båda metoderna. Se även REF 3 sid 143.

Vi tänker oss en funktion $F(X)$, där $X=(x_1,x_2, \ldots , x_n)$, dvs en funktion av n variabler.

Vi vill finna ett lokalt minimum till funktionen. Med lokalt minimum menas populärt uttryckt, att vi har uppskattat minimipunkten till $U = (u_1, u_2, \ldots, u_n)$, och vill nu beräkna en bättre uppskattning, dvs beräkna en riktning i vilken funktionen avtar. Den nya uppskattningen är $U+S = (u_1+s_1, u_2+s_2, \ldots, u_n+s_n)$.

I minimipunkten är $\left[\frac{d}{dx_i} F(X)\right]_{X=U+S} = 0 \quad , i=1,2, \ldots, n.$

Vi Taylor-utvecklar uttrycket kring punkten U och tar med två termer.

$$\left[\frac{d}{dx_i} F(X)\right]_{X=U} + \sum_{j=1}^{n} \left[\frac{d^2}{dx_i dx_j} F(X)\right]_{X=U} \cdot s_j = 0 \quad , \quad i=1, 2, \ldots, n$$

Beteckningar:

$$g_i = \left[\frac{d}{dx_i} F(X)\right]_{X=U}$$

$$G_{ij} = \left[\frac{d^2 F(X)}{dx_i dx_j}\right]_{X=U}$$

$$g = (g_1, g_2, \ldots, g_n)$$

$$S = (s_1, s_2, \ldots, s_n)$$

$$G = \begin{pmatrix} G_{11} & G_{12} & \cdots & G_{1n} \\ G_{21} & \cdots & \cdots & \\ \vdots & \vdots & & \\ G_{n1} & \cdots & \cdots & G_{nn} \end{pmatrix}$$

Då fås $g + G \cdot S = 0$

$S = -G^{-1} g$, där S är minimeringsriktningen.

Vår nästa uppskattning av minimipunkten blir därför

$$U_{(k+1)} = U_{(k)} - \varepsilon \cdot G^{-1}_{(k)} \, g_{(k)}$$

Där (k) betecknar den gamla punkten och (k+1) den nya. ε är steglängden i riktningen S.

S är alltså riktningen, man skall förflytta sig i för att F skall bli mindre. Det är emellertid uppenbart, att om man fortsätter att gå i riktningen S, så kommer F till slut att öka för de funktioner vi sysslar med här. Det verkar som om man behöver minimera linjärt längs riktningen S för att vara säker på, att man tar ett effektivt steg. Vi återkommer till det problemet i avsnittet om stabilitet.

För att kunna minimera funktionen måste dess gradient och andraderivatmatris beräknas i varje iteration.

3. Metodernas särdrag

Andraderivatmatrisen G tas fram på olika sätt i de båda metoderna.

I Newton-Raphson beräknas på sedvanligt sätt G, som därefter invertas.

I Fletcher-Powell- metoden användes en matris H, som motsvarar G^{-1}. H sättes från början till enhetsmatrisen, vilket medför att första steget tas i riktning "steepest descent". Sedan beräknas H genom ett iterationsförfarande, som står beskrivet i REF 2. Man behöver således inte invertera någon matris. Då man nått minimipunkten är H en god approximation av G^{-1}.

I REF 2 användes Diracś beteckningar för vektorer:

Kolonnvektorn x betecknas med $|x\rangle$
Radvektorn x betecknas med $\langle x|$
Skalärprodukten av xoch y betecknas med $\langle x|y\rangle$
Om H är en matris, är $\langle x|H$ en radvektor
osv

4. Metodernas stabilitet

Det är naturligt att fråga sig om minimeringsmetoderna alltid är stabila.

I fallet Newton-Raphson är det lättast att svara. Förutsatt att man startar tillräckligt nära minimipunkten visar härledningen, att metoden alltid bör vara stabil. Man beräknar ju gradienten g och andraderivatmatrisen G i varje punkt, vilket medför att man alltid kan finna ett mindre funktionsvärde längs riktningen S. Metoden är alltid stabil, om man väljer steglängden så, att varje funktionsvärde blir mindre än det föregående.

Villkoren för stabilitet hos Fletcher- Powells metod är utredda i REF 2, avsnitt 4. Jag gör en sammanfattning här.

Gradienten $|g\rangle$ pekar alltid mot brantast ökande funktionsvärde. $|S\rangle$ pekar mot minskande funktionsvärde om och endast om $-\langle g|S\rangle$ är positivt. ($\langle g|S\rangle$ är riktningsderivatan i riktningen $|S\rangle$). Men $|S\rangle = -H|g\rangle$, dvs $|S\rangle$ pekar mot minskande funktionsvärde om och endast om $\langle g|H|g\rangle$ är positivt. Detta är detsamma som att matrisen H är positivt definit.

Villkoret för stabilitet är alltså att H är positivt definit i alla iterationerna. Enhetsmatrisen är pos def, varför H är pos def från början. I avsnitt 4 i REF 2 kommer man fram till att H förblir pos def om uttrycket

$$\langle s_{(k)} \mid g_{(k+1)} \rangle - \langle s_{(k)} \mid g_{(k)} \rangle$$

är positivt. Med ord kan det uttryckas: Riktningsderivatan i riktningen S i den nya punkten skall vara mindre negativ (dvs ha mindre abs-värde) än i den gamla punkten.

En förtydligande figur:

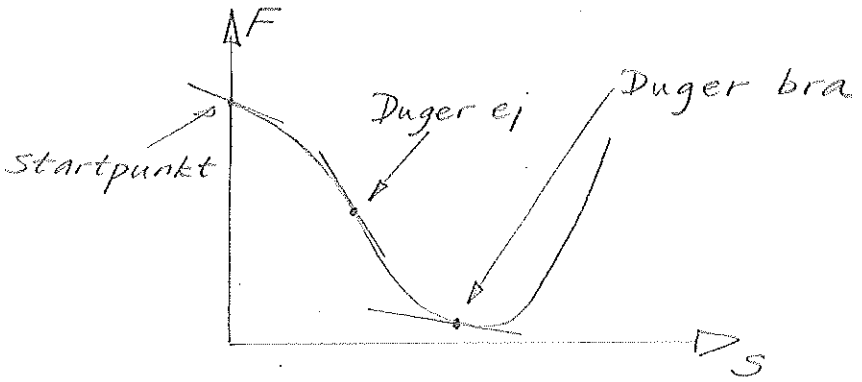


Fig 2.1.

Om den nya punkten väljes något förbi minimipunkten, är metoden självklart stabil om funktionen har ett utseende enligt fig 2.1, ty

$$\underbrace{\langle s_{(k)} \mid g_{(k+1)} \rangle}_{A} - \underbrace{\langle s_{(k)} \mid g_{(k)} \rangle}_{B}$$

både A och B är då positiva. Man inser då att Fletcer-Powells metod kräver en noggrann linjär minimeringsmetod för att vara stabil.

Newton-Raphsons metod däremot är alltid stabil, men där kan man råka ut för att G blir singulär. Även om funktionen skulle ha singulär andraderivatmatris i minimipunkten, så fungerar Fletcher-Powells metod ändå.

KAP 3. BERÄKNING AV FÖRLUSTFUNKTIONEN, DESS GRADIENT OCH ANDRADERIVATMATRIS

1. Inledning

Då man skall beräkna förlustfunktionen måste man avgöra vilken representation av ekvationerna man skall välja, den kontinuerliga eller den samplade formen.

Den samplade formen ger mindre räknetid och den har man använt i REF 1. Eftersom det nu på institutionen finns en effektiv subrutin för att exponentiera en matris, valde jag den kontinuerliga formen.

Fördelen är då, att man får ut överföringsfunktionen direkt utan några omräkningar.

I programmet utföres beräkningen av förlustfunktion, gradient och andraderivata i subrutinen VPRIMBIS.

2. Version 1

Förlustfunktionen V

Om ett dynamiskt system

$$\begin{cases} \dfrac{dx}{dt} = Ax + Bu \\ YB = Cx \end{cases} \qquad (3.1)$$

är observerbart kan det alltid skrivas på den observerbara kanoniska formen, med

$$A = \begin{bmatrix} -a_1 & 1 & . & 0 \\ -a_2 & 0 & 1 & 0 \\ . & . & . & 1 \\ -a_N & 0 & . & 0 \end{bmatrix} \qquad B = \begin{bmatrix} b_1 \\ b_2 \\ . \\ b_N \end{bmatrix}$$

$$C = (1, 0, \ldots., 0)$$

I version 1 användes denna formen på ekvationerna.

Med en Dirac-puls som insignal kan ekvationerna skrivas

$$\begin{cases} \dfrac{dx}{dt} = Ax \\ x(0) = B \\ YB = Cx \end{cases}$$

Lösningen blir $\begin{cases} x(t) = e^{At} B \\ YB(t) = Cx \end{cases}$

Den beräknade utsignalens värde i mätpunkt nr k är $YB_k = YB(t)$, med $t = (k-1)T$ där T är tiden mellan två avläsningar. Enligt ovan blir YB(t) skalärprodukten av första raden i e^{At} och B.

Förlustfunktionen $V = \sum_{k=1}^{M} (YB_k - YM_k)^2$ kan nu beräknas eftersom YM_k förutsättes vara känd.

Gradienten GRADV

$$V = \sum_{k=1}^{M} (YB_k - YM_k)^2$$

Inför en vektor $X = (x_1, x_2, \ldots , x_{2N}) =$

$$= (a_1, a_2, \ldots, a_N, b_1, b_2, \ldots , b_N)$$

YB_k är alltså en funktion av X. Deriveras V partiellt med avseende på x_i erhålles

$$\frac{dV}{dx_i} = 2 \sum_{k=1}^{M} (YB_k - YM_k) \frac{dYB_k}{dx_i}$$

Vi måste då beräkna derivatan

$$\frac{dYB_k}{dx_i} = \frac{dYB(t)}{dx_i} = \frac{d}{dx_i} (Ce^{At} B)$$

Vi delar upp i två fall :

1. $N < i \leq 2N$

$x_{N+1}, x_{N+2}, \ldots , x_{2N}$ återfinns i matrisen B, så derivatan blir

$$\frac{dYB(t)}{dx_i} = Ce^{At} \frac{dB}{dx_i} = \text{i:te elementet i första raden i } e^{At}.$$

För t=0 blir $\frac{dYB(0)}{dx_{N+1}} = 1$, de andra blir lika med noll.

2. $1 \leq i \leq N$

$x_1, x_2, \ldots, x_N$ återfinns i matrisen A.

$$\frac{dYB(t)}{dx_i} = C \frac{de^{At}}{dx_i} B$$

OBS $\frac{de^{At}}{dx_i} \neq t \cdot e^{At} \cdot \frac{dA}{dx_i}$

ty e^{At} och $\frac{dA}{dx_i}$ kommuterar ej.

I stället inför vi en kvadratisk matris Y, som uppfyller

$\frac{dY}{dt} = A\,Y$ med $Y(0)=I$, dvs $Y(t) = e^{At}\,I$, där I är enhetsmatrisen. Derivera $\frac{dY}{dt} = AY$ med avseende på x_i

$$\frac{d}{dx_i}\frac{dY}{dt} = A \frac{dY}{dx_i} + \frac{dA}{dx_i} Y$$

Byt derivationsordning och sätt $Z = \frac{dY}{dx_i}$ dvs Z är vår sökta derivatmatris.

$$\frac{dZ}{dt} = AZ + \frac{dA}{dx_i} Y$$

Lös i stället diff.-ekvationssystemet

$$\begin{cases} \frac{dY}{dt} = AY \\ \frac{dZ}{dt} = \frac{dA}{dx_i} Y + AZ \end{cases}$$

eller på matrisform

$$\frac{d\begin{pmatrix} Y \\ Z \end{pmatrix}}{dt} = \begin{bmatrix} A & 0 \\ \frac{dA}{dx_i} & A \end{bmatrix} \begin{bmatrix} Y \\ Z \end{bmatrix}$$

med $\begin{pmatrix} Y(0) \\ Z(0) \end{pmatrix} = \begin{pmatrix} I \\ 0 \end{pmatrix}$

Diff.-ekvationen har lösningen

$$\begin{pmatrix} Y \\ Z \end{pmatrix} = e^{\begin{pmatrix} A & 0 \\ \frac{dA}{dx_i} & A \end{pmatrix} t} \cdot \begin{pmatrix} I \\ 0 \end{pmatrix} \qquad \text{Ekv. 3.2}$$

e^{At} fås i matrisens övre halva och

$\frac{de^{At}}{dx_i}$ fås i matrisens undre halva.

Vi har nu beräknat derivatan vid tidpunkten t, och vill beräkna derivatan vid t+T utan att utföra ny exponentiering.

$$Y(t+T) = Y(t)\ e^{AT}$$

$$Z(t+T) = \frac{dY(t+T)}{dx_i} = \frac{dY(t)}{dx_i}\ e^{AT} + Y(t)\ \frac{de^{AT}}{dx_i}$$

$$= Z(t)\ e^{AT} + Y(t)\ Z(T) \qquad \text{Ekv. 3.3}$$

Vi sammanfattar resultatet för $1 \leq i \leq N$

$t = 0$ $\quad \frac{dYB(0)}{dx_{N+1}} = 1$, alla övriga derivator noll

$t = T$ $\quad \frac{dYB(T)}{dx_i} = C\ \frac{de^{AT}}{dx_i}\ B = C\ Z(T)\ B$

där $Z(T)$ fås ur ekv. 3.2.

$t = 2T$ $\quad$ ekv. 3.3 ger $Z(T+T) = Z(T)\ e^{AT} + Y(T)\ Z(T)$

osv

Vi ser att man måste utföra exponentieringen 2 gånger för ett andra ordningens system, 3 gånger för ett tredje ordningens osv.

Detta beräkningssätt blir mindre attraktivt ju högre ordningstal man räknar med. I Version 2 beskrives ett annat beräkningssätt, som är smidigare.

Andraderivatmatrisen G

Derivera $$\frac{dV}{dx_i} = 2 \sum_{k=1}^{M} (YB_k - YM_k) \frac{dYB_k}{dx_i}$$

$$\frac{d^2V}{dx_j dx_i} = 2 \sum_{k=1}^{M} (YB_k - YM_k) \frac{d^2YB_k}{dx_j dx_i} + 2 \sum_{k=1}^{M} \frac{dYB_k}{dx_i} \frac{dYB_k}{dx_j}$$

Enligt REF 3 sid 148 kan man approximera uttrycket med enbart sista termen

$$\frac{d^2V}{dx_j dx_i} = 2 \sum_{k=1}^{M} \frac{dYB_k}{dx_i} \cdot \frac{dYB_k}{dx_j}$$

Beräkningen blir då enkel. Man måste emellertid lagra $\frac{dYB_k}{dx_i}$ för alla k och i.

3. Version 2

Vi skall nu med hjälp av teorin för dynamiska system försöka komma fram till en enklare metod för att beräkna GRADV.

Insignalen är fortfarande en Dirac-puls. Jag beskriver beräkningen för ett andraordningens system. Utvidgningen är självklar.

Vi använder nu representationen

$$\begin{cases} \dfrac{dx}{dt} = Ax \\ \\ YB = b_1x_1 + b_2x_2 \end{cases}$$

med $A = \begin{pmatrix} -a_1 & -a_2 \\ 1 & 0 \end{pmatrix}$ och $x(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

Eftersom Laplace-transformen för en Dirac-puls är en etta blir utsignalen i Laplace-transform

$$YB(s) = \frac{b_1s+b_2}{s^2+a_1s+a_2}$$

Derivera YB(s) med avseende på systemparametrarna, så fås

$$\frac{dYB(s)}{da_1} = -\frac{s}{s^2+a_1s+a_2}\,YB(s) \qquad \frac{d\,YB(s)}{db_1} = \frac{s}{s^2+a_1s+a_2}$$

$$\frac{dYB(s)}{da_2} = -\frac{1}{s^2+a_1s+a_2}\,YB(s) \qquad \frac{dYB(s)}{db_2} = \frac{1}{s^2+a_1s+a_2}$$

Sätt $z_1 = \frac{dYB}{da_1}$ $z_3 = \frac{dYB}{db_1}$

$z_2 = \frac{dYB}{da_2}$ $z_4 = \frac{dYB}{db_2}$

dvs $z_1 = -\frac{s}{s^2+a_1s+a_2}\,YB$ $z_3 = \frac{s}{s^2+a_1s+a_2}$

$z_2 = -\frac{1}{s^2+a_1s+a_2}\,YB$ $z_4 = \frac{1}{s^2+a_1s+a_2}$

Skriv om som diff.-ekvation

$$\begin{cases} \frac{dz_1}{dt} + a_1z_1 + a_2z_2 = -b_1z_3 - b_2z_4 \\ \frac{dz_2}{dt} = z_1 \\ \frac{dz_3}{dt} + a_1z_3 + a_2z_4 = 1 \\ \frac{dz_4}{dt} = z_3 \end{cases}$$

ty $YB(s) = \frac{b_1s+b_2}{s^2+a_1s+a_2} = b_1z_3 + b_2z_4$

eller på matrisform $\frac{dz}{dt} = A'z$

med $A' = \begin{pmatrix} -a_1 & -a_2 & -b_1 & -b_2 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -a_1 & -a_2 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ och $z(0) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$

Diff.-ekvationen har lösningen $z(t) = e^{A't} z(0)$.

Vi har då beräknat alla gradienterna med bara en exponentiering av matrisen. Om man enbart vill beräkna V, behöver man bara exponentiera den övra vänstra kvadranten av matrisen.

För övrigt beräknas V och G på samma sätt som i Version 1.

KAP 4. DE LINJÄRA MINIMERINGSMETODERNA

1. Inledning

Som vi sett i kap. 2, spelar den linjära minimeringen en betydande roll vid Fletcher-Powells metod.

Jag har undersökt tre linjära minimeringsmetoder. I programmet återfinns de i var sin subrutin, nämligen LINMINA, LINMINB och LINMINC.

2. Linjär minimering LINMINA

Beräkningen i LINMINA bygger på en metod given i appendix i REF 2. I beräkningen används funktionsvärdet och gradienten.

Principen är följande: Man tar ett steg med längden ETA i minimeringsriktningen. Med hjälp av funktionsvärde och gradient i begynnelsepunkten (x) och i den nya punkten (y) beräknas sedan det linjära minimet. Beräkningen följer exakt REF 2.

Problemet är då hur man skall välja ETA. I REF 2 väljes

$$ETA = \min\left(1 \ , \ \frac{-2(fx-fo)}{\langle gx \mid S \rangle}\right)$$

där fx är funktionsvärdet i (x)
fo är funktionens minsta värde
(jag har satt fo=0)
$\langle gx|S \rangle$ är skalärprodukten av minimeringsriktningen och gradienten i (x)

Vid körningar på system med ordningstalet N större än två, visade det sig, att ETA blev för litet. I programmet användes därför uppskattningen

$$ETA = \min\left(1 \ , \ \frac{-2 \cdot N \cdot fx}{\langle gx | S \rangle}\right)$$

Det visade sig också nödvändigt att ta ytterligare steg framåt, tills $\langle gy | S \rangle$ blev positivt, dvs man fick stega sig fram tills det linjära minimet var passerat. Vid varje stegning fördubblas då ETA.

3. Linjär minimering LINMINB

Denna minimeringsmetod är en mycket enkel sådan och är närmast avsedd att användas i samband med Newton-Raphsons minimeringsmetod.

Principen är följande: Man tar ett steg ETA=1 i minimeringsriktningen. Om det nya funktionsvärdet är mindre än det gamla, nöjer man sig med den punkten. Annars halveras steget tills man nått ett funktionsvärde, som är mindre än det ursprungliga.

4. Linjär minimering LINMINC

Den här metoden, som vi kan kalla Fibbonacci-metoden, finns återgiven i REF 4. Vid beräkningen använder man enbart funktionsvärdet.

Detta kan vara attraktivt, eftersom gradientberäkningen för en förlustfunktion till ett dynamiskt system är en tämligen komplicerad procedur, som vi såg i kap.3.

Principen är följande: Antag att man vet att minimet ligger mellan punkterna (x) och (y). Då väljer man två punkter (v) och (h), som ligger mellan (x) och (y), se fig 4.1. Funktionsvärdena f_v och f_h beräknas.

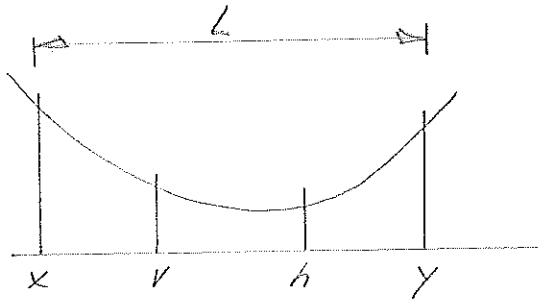


Fig 4.1

Om $f_v \leq f_h$ ligger minimet mellan (x) och (h) annars mellan (v) och (y). Man har då förminskat det ursprungliga intervallet. Sedan gör man samma sak tills intervallet är tillräckligt litet.

Problemet är då hur man skall välja punkterna (v) och (h). Här kommer Fibbonacci- talen in i bilden. Fibbonacci-talen definieras sålunda: $F_1=1$, $F_2=1$, $F_n = F_{n-1} + F_{n-2}$.

Då n blir stort gäller $\frac{F_{n-1}}{F_{n+1}} = 0.382$ och $\frac{F_n}{F_{n+1}} = 0.618$.

Om intervall-längden är L, så väljes (v) och (h) på ett optimalt sätt om

$$v = x + 0.382\ L$$
$$h = x + 0.618\ L$$

med beteckningar enligt fig 4.1. Finessen är då, att man behöver bara beräkna ett funktionsvärde för varje gång. Om $f_v \leq f_h$ så väljes den gamla vänsterpunkten som ny högerpunkt och en ny vänsterpunkt beräknas och vice versa.

På 5 iterationer förminskas intervallet till 1/10 och på 10 iterationer till 1/100 av det ursprungliga.

Hur skall man då bestämma det ursprungliga intervallet inom vilket minimet ligger? Jag har gjort på följande sätt: Först kontrolleras att man med steget ETA kan hitta ett funktionsvärde, som är mindre än det ursprungliga. Annars sättes ETA:= 0.1·ETA . Från begynnelsepunkten (x) stegar man sig sedan fram, tills man hittar en punkt (y) med funktionsvärdet större än det föregående. Då vet man att minimet ligger inom intervallet I = (y - 2ETA , y). Se fig 4.2.

Som steglängd används samma ETA som beräknades i LINMINA. Om inte minimet är lokaliserat efter 3 stegningar fördubblas ETA för varje gång.

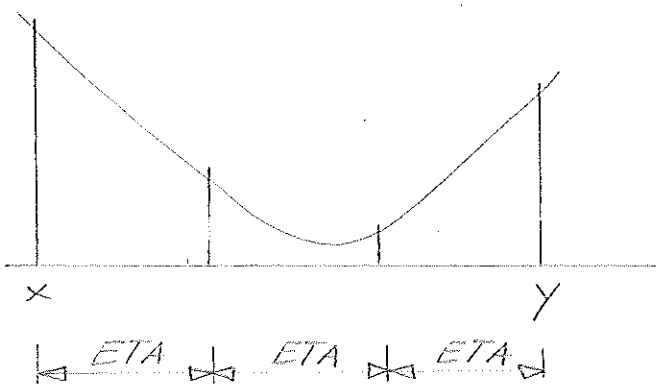


Fig 4.2

KAP 5. PROGRAMMET

1. Inledning

Programmet är skrivet i FORTRAN, och körningarna har skett från en terminal i Lund på Uppsala-maskinen CD 3600. Maskinen är mycket snabb. Räknetider: flytande addition 4.25 mikrosek, flytande multiplikation 6.00 mikrosek.

FORTAN-språket, som använts här skiljer sig något från den normala versionen. Bl a får man ha 8 positioner i variabelnamnen, och utskriftssatserna är olika.

2. Programmets uppbyggnad

Eftersom programmet från början såg ut att bli tämligen omfattande, bestämde jag mig för att dela upp det i många subrutiner.

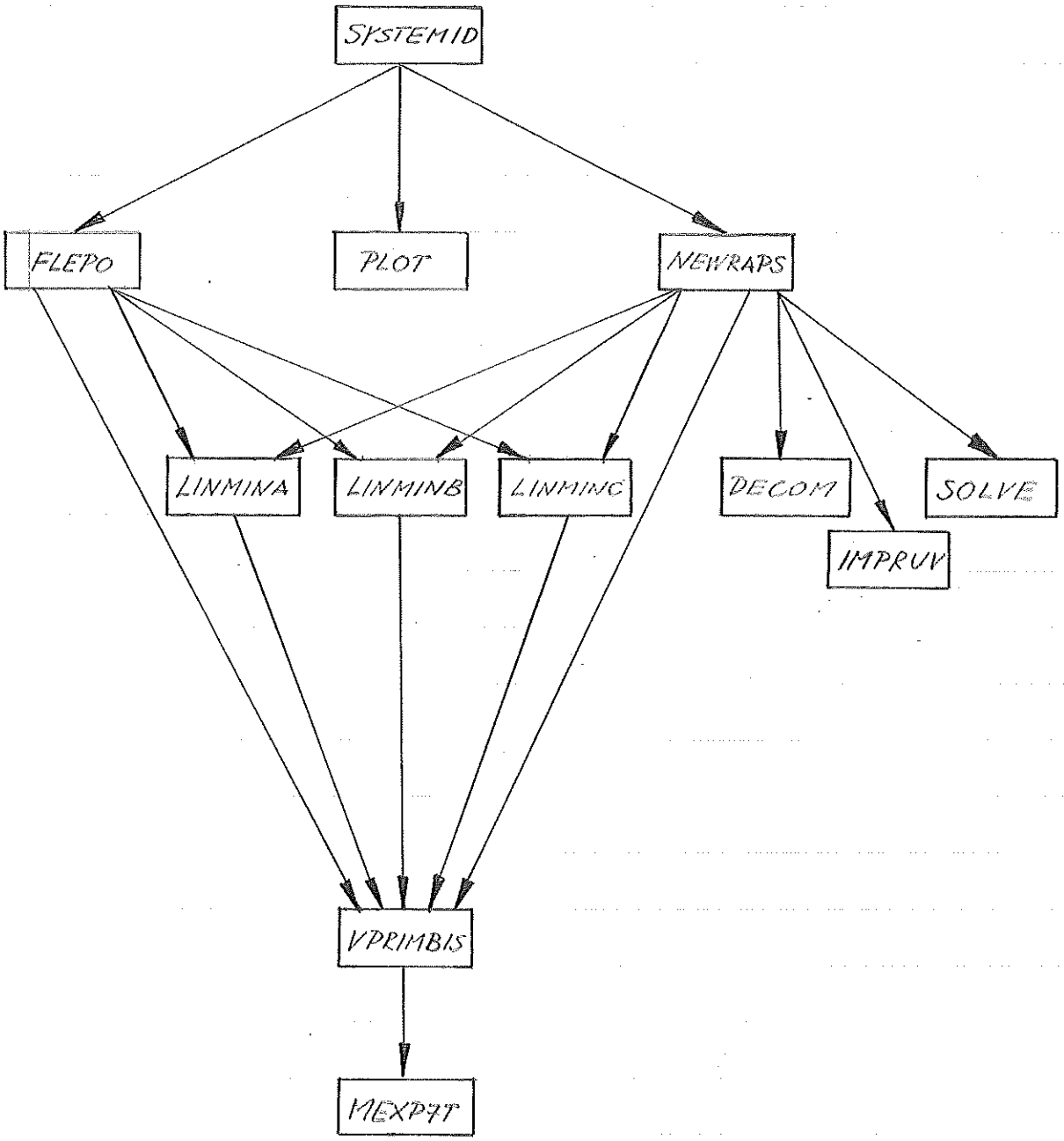
Då blir programmet mera flexibelt, dvs det kan användas för många ändamål med små förändringar t ex genom att byta ut någon subrutin. En annan fördel är att man kan testa varje subrutin för sig och därigenom lättare lokalisera fel i programmet.

På nästa sida följer en förteckning över huvudprogrammet och subrutinerna samt deras funktion.

PROGRAM SYSTEMID	Inläsning och administration
SUBROUTINE FLEPO	Minimering med Fletcher-Powells metod
SUBROUTINE NEWRAPS	Minimering med Newton-Raphsons metod
SUBROUTINE LINMINA	Linjär minimering enligt REF 2
SUBROUTINE LINMINB	Enkel form av linjär minimering
SUBROUTINE LINMINC	Linjär minimering med Fibbonacci-tal
SUBROUTINE VPRIMBIS	Beräkning av förlustfunktion, gradient och andraderivatmatris
SUBROUTINE DECOM	Författare Per Hagander
SUBROUTINE SOLVE	Lösning av ekvationssystem
SUBROUTINE IMPRUV	
SUBROUTINE MEXP7T	Författare Krister Mårtensson Exponentiering av matris
SUBROUTINE PLOT	Plottning av två funktioner av en oberoende variabel på radskrivare

För att klara ut hur anropen kan ske, ställer jag upp följande schema, fig 5.1.

betyder att subrutin B anropas från program eller subrutin A . Återhoppet är ej utritat.



3. PROGRAM SYSTEMID.

I huvudprogrammet sker följande:

1. Mätdata och begynnelsedata läses in, Se även avsnitt 5.5.

2. Utskrift av minimeringsmetod, antal mätpunkter, tidsintervallet mellan två mätpunkter STEGTID, ordningstal N och begynnelseparametervärdena STARTX(I).

3. FLEPO eller NEWRAPS anropas och resultatet efter minimeringen plottas i PLOT.

4. Man undersöker om förlustfunktionen är tillräckligt liten. Om inte, så höjes ordningstalet. Man går tillbaka till punkt 3 med de erhållna parametervärdena. X(N)=X(2N)=0.

   Det är lämpligt att börja med N=2. Då blir X(N+1)= YM(0) och man gissar de tre övriga parametrarna. Gissningen kan utföras mycket approximativt.

   Om MHALV läses in lika med 1, så användes först bara varannan mätpunkt för en "grovräkning". När förlustfunktionen V blir tillräckligt litet, användes alla mätpunkterna vid samma ordningstal.

   All kommunikation mellan program och subrutiner och mellan subrutiner sker med Common-fältet. Commonvariablerna förklaras i programutskriften.

4. Kommentarer till subrutinerna NEWRAPS och FLEPO

Uthopp ur subrutinerna sker då absolutbeloppet för riktningsvektorn $|S\rangle$ är mindre än EPS1. Uthopp ur FLEPO sker tidigast efter 2N iterationer.
Iterationsantalet är begränsat till 100. Om detta uppnås, fås utskriften IANT=100 .

I FLEPO testas att riktningen $|S\rangle$ pekar mot minskande funktionsvärde. Om så ej är fallet, sättes H till enhetsmatrisen och man går då i riktning "steepest descent".
Utskrift:MATRIS NEG DEF .

I NEWRAPS löses ekvationssystemet $G|S\rangle = -|g\rangle$ i stället för att invertera G och sedan multiplicera med $|g\rangle$.

Om G blir singulär, så ändras x-värdena något, (X(I)= 1.02X(I)) och minimeringsriktningen $|S\rangle$ beräknas på nytt.
Utskrift: G SINGULAR . Hela G-matrisen skrives också ut.

5. Inläsning och utskrifter

Variablerna som läses in förklaras i utskriften av PROGRAM SYSTEMID, där all inläsning sker.

Inläsningen sker på följande sätt:

		FORMAT
Kort 1	N	I5
	M	I5
	METOD	I5
	METODOPT	I5
	STEGTID	F10.5
	EPS1	F10.5
Kort 2	MHALV	I5
	ISYMM	I5
	SKALFAK	F10.5
	SLUTV	F10.5
Kort 3	a_1	F10.5
och om N>4	a_2	"
även kort 4	.	"
	.	"
	a_N	"
	b_1	"
	b_2	"
	.	"
	.	"
	b_N	"
Nästa kort		
och följande	YM_1	F10.5
	YM_2	"
	.	"
	.	"
	YM_M	"

Utskrift av förlustfunktionens värde och parametervärdena sker från subrutinerna FLEPO och NEWRAPS enligt följande mall:

		a_1	a_2	a_3	.	.	a_N
V		b_1	b_2	b_3	.	.	b_N

Utskrift av YM_i och YB_i sker från subrutinen PLOT.

6. Programutskrift

FTN5.48 02/04-69

```
      PROGRAM SYSTEMID
C
C     AUTHOR BERTIL LUNDGREN  MARCH 1969
C     IDENTIFIES A DETERMINISTIC DYNAMIC LINEAR SYSTEM
C     OF ORDER LESS OR EQUAL TO 10
C     REQUIRED SUBROUTINES  FLEPO NEWRAPS PLOT
C
C     EXPLANATION OF READ-VARIABLES
C     N  IS ORDER OF THE SYSTEM
C     M IS THE NUMBER OF MEASUREPOINTS
C     IF METOD=1 THEN LINMINA IS USED
C     IF METOD=2 THEN LINMINB IS USED
C     IF METOD=3 THEN LINMINC IS USED
C     IF METODOPT=1 THEN FLEPO IS USED
C     IF METODOPT=2 THEN NEWRAPS IS USED
C     STEGTID  IS THE TIME BETWEEN TWO MEASUREPOINTS
C     EPS1  IS THE ACCURACY USED IN FLEPO AND NEWRAPS
C     DELTA  IS RESERV-VARIABLE, NOT USED
C     IF MHALV=1 EVERY SECOND MEASUREPOINT IS USED AT FIRST
C     IF MHALV=2 ALL THE MEASUREPOINTS ARE USED AT ONCE
C     ISYMM AND SKALFAK ARE EXPLAINED IN PLOT
C     WHEN SLUTV IS LESS THAN V, COMPUTATION IS FINISHED
C     STARTX(I)  ARE THE SYSTEMPARAMETERS AT THE BEGINNING
C     STARTX(1),,,,STARTX(N) CORRESPOND TO A(1),,,,A(N)
C     STARTX(N+1),,,,STARTX(N2) CORRESPOND TO B(1),,,,B(N)
C     YMA(I)  IS THE MEASURED VALUE OF SYSTEM-OUTPUT IN
C             MEASUREPOINT NUMBER I
C
C     EXPLANATION OF COMMON-ARRAY
C     X(I) ARE THE SYSTEMPARAMETERS, SEE STARTX(I) ABOVE
C     N2  IS THE NUMBER OF VARIABLES
C     M STEGTID AND METOD  ARE EXPLAINED ABOVE
C     IF K=0 ONLY V IS COMPUTED IN VPRIMBIS
C     IF K=1 V AND GRADV ARE COMPUTED IN VPRIMBIS
C     IF K=2  V GRADV AND G ARE COMPUTED IN VPRIMBIS
C     V  IS VALUE OF LOSS-FUNCTION
C     GRADV(I)  IS GRADIENT OF V
C     G(I,J)  IS SECOND-DERIVATIVEMATRIX OF V
C     S(I) IS DIRECTION OF MINIMIZATION
C     IANT  IS NUMBER OF ITERATIONS PERFORMED IN FLEPO AND NEWRAPS
C     SGRADV  IS SCALARPRODUCT BETWEEN S(I) AND GRADV(I)
C     EPS1  IS EXPLAINED ABOVE
C     YM(I) ARE THE MEASUREPOINTS USED
C     YB(I)  ARE THE CALCULATED CORRESPONDINGS TO YM(I)
C
      DIMENSION STARTX(20), YMA(200)
      COMMON X(20), N2, M, STEGTID, METOD,  K, V, GRADV(20), G(20,20),
     F S(20), DELTA,  IANT, SGRADV, EPS1, YM(200),YB(200)
      DATA (A= 1HA),(B= 1HB),(C=1HC)
      INTEGER A,B,C
      READ 105, N, M, METOD, METODOPT, STEGTID, EPS1, DELTA
      READ 104, MHALV, ISYMM, SKALFAK, SLUTV
      N1= N + 1
      N2= 2*N
      READ 106, ( STARTX(I), I= 1, N2 )
      READ 106, ( YMA(I), I= 1, M )
```

```
C
      IF ( METOD .EQ. 1 ) MPRINT= A
      IF ( METOD .EQ. 2 ) MPRINT= B
      IF ( METOD .EQ. 3 ) MPRINT= C
   02 DO 03 I= 1, N2
   03 X(I)= STARTX(I)
C
   04 GO TO (05,08), METODOPT
   05 PRINT 107, MPRINT
      GO TO 09
   08 PRINT 108, MPRINT
C     IF MHALV=1 THEN EVERY SECOND MEASUREPOINT IS USED AT FIRST
   09 GO TO ( 10, 20 ), MHALV
   10 M= M/2
      DO 15  I= 1, M
      J= 2*I - 1
   15 YM(I)= YMA(J)
      STEGTID= 2.*STEGTID
      GO TO 30
   20 DO 25 I= 1, M
   25 YM(I)= YMA(I)
   30 PRINT 110, M, STEGTID
C
   50 PRINT115, N, (      X(I), I= 1,N )
      PRINT 116, (       X(I), I= N1, N2 )
      MS= MSLEFT(1)
      PRINT 180, MS
  180 FORMAT ( 34HMS=, I7 )
      GO TO ( 55, 60 ), METODOPT
   55 CALL FLEPO
      GO TO 62
   60 CALL NEWRAPS
   62 CALL PLOT ( YM,YB,M,ISYMM, SKALFAK )
      IF (  V - SLUTV  )  80, 80, 65
   65 IF ( N - 9 ) 70, 70, 90
   70 DO 72 I= 1, N
      J= N2 + 1 - I
   72 X(J+1)= X(J)
      N= N + 1
      N1= N + 1
      N2= 2*N
      X(N )= 0.0
      X(N2)= 0.0
      GO TO 50
C
   80 GO TO ( 85, 95 ), MHALV
   85 MHALV= 2
      STEGTID= 0.5*STEGTID
      M= 2*M
      GO TO 20
   90 PRINT 165
      GO TO 98
   95 PRINT 175
   98 CONTINUE
C
```

FTN5.48 02/04-09

```
  104 FORMAT ( 2I5,7F10.5 )
  105 FORMAT ( 4I5, 3F10.5 )
  106 FORMAT ( 8F10.5 )
  107 FORMAT(/10X,34HOPT-METOD FLETCHER-POWELL    LINMIN,A1/10X,35(1H*))
  108 FORMAT(/10X,34HOPT-METOD NEWTON-RAPHSON     LINMIN,A1/10X,35(1H*))
  110 FORMAT(////16HANT MAETPUNKTER=,I3,12H    STEGTID=,F7.3//)
  115 FORMAT (//4H  N=,I2/6H  ****//11HSTARTVEKTOR, (10F10.4))
  116 FORMAT ( 11X, (10F10.4))
  165 FORMAT ( /10X, 22HSYST-ORDNINGSTAL = 10 )
  175 FORMAT (/10X, 7HHEUREKA )
      CALL EXIT
      END
```

FTN5.4B 02/04-69

```
      SUBROUTINE FLEPO
C
C     REFERENCE COMPUTER-JOURNAL VOL.6 1963
C     AUTHOR BERTIL LUNDGREN  MARCH 1969
C     MINIMIZATION OF FUNCTION OF N2 VARIABLES, STARTING AT X(I)
C     METHOD  FLETCHER-POWELL
C     X(I) IS GIVEN THE COORDINATE OF THE MINIMUM
C     COMMON-ARRAY IS EXPLAINED IN SYSTEMID
C     INDATA  X(I) N2 METOD EPS1
C     OUTDATA  X(I)
C
C     REQUIRED SUBROUTINES  LINMINA LINMINB LINMINC VPRIMBIS
C
      DIMENSION OLDGRADV(20), H(20,20), A(20,20), B(20,20), SIGMA(20),
     F Y(20), HGGRY(20), YGGRH(20), OLDX(20)
      COMMON X(20), N2, M, STEGTID, METOD,  K, V, GRADV(20), G(20,20),
     F S(20), DELTA,  IANT, SGRADV, EPS1, YM(200),YB(200)
      N= N2/2
      N1= N + 1
      K= 1
      IANT=0
C
C     H IS SET TO UNITY-MATRIX
   08 DO 10 I=1, N2
      DO 10 J=1, N2
   10 H(I,J)=0.
      DO 15 I=1, N2
   15 H(I,I)=1.
C
      CALL VPRIMBIS
   18 IANT= IANT+1
      DO 20  I= 1, N2
      OLDX(I)= X(I)
   20 OLDGRADV(I)= GRADV(I)
C
C     COMPUTE NEW DIRECTION S(I)
      DO 22 I= 1, N2
         S(I)= 0.
      DO 22 J= 1, N2
   22     S(I)= S(I)-H(I,J)*GRADV(J)
C
C     TEST IF GOING DOWNHILL
      SGRADV= 0.
      DO 122 I= 1, N2
  122 SGRADV= SGRADV + S(I)*GRADV(I)
      IF ( SGRADV ) 130, 28, 124
  124 PRINT 128
  128 FORMAT ( 13HMATRIS NEGDEF )
      GO TO 08
C
C     LINEAR MINIMIZING
  130 GO TO (23, 24, 25), METOD
   23 CALL LINMINA
      GO TO 26
   24 CALL LINMINB
      GO TO 26
```

FTN5.4B 02/04-69

```
      25 CALL LINMINC
      26 DO 27 I= 1, N2
      27 SIGMA(I)= X(I) - OLDX(I)
      28 PRINT 29, ( X(I), I= 1, N )
         PRINT 30, V, ( X(I), I= N1, N2 )
      29 FORMAT (/19X, 10F10.4 )
      30 FORMAT ( 9X, E10.3, 10F10.4 )
C
C     TEST IF COMPUTATION WILL BE FINISHED
         IF ( IANT -100 ) 33, 33, 51
      33 IF ( IANT-N2 ) 35, 35, 31
      31 SKALS= 0.
         SKALSIG= 0.
         DO 132 I= 1, N2
         SKALSIG= SKALSIG + SIGMA(I)*SIGMA(I)
     132 SKALS= SKALS + S(I)*S(I)
         EPSKVA= EPS1*EPS1
         IF (SKALS - EPSKVA) 34, 35, 35
      34 IF ( SKALSIG - EPSKVA ) 52, 35, 35
      35 CONTINUE
C
C     COMPUTE NEW H-MATRIX
         DO 36 I= 1, N2
      36    Y(I)= GRADV(I)- OLDGRADV(I)
         YHY= 0.
         DO 40 I= 1, N2
            HGGRY(I)=0.
            DO 38 J= 1, N2
      38    HGGRY(I)= HGGRY(I)+ H(I,J)*Y(J)
      40 YHY= YHY+HGGRY(I)*Y(I)
         DO 42 J= 1, N2
            YGGRH(J)= 0.
            DO 42 I= 1, N2
      42 YGGRH(J)= YGGRH(J)+Y(I)*H(I,J)
         DO 46 I= 1, N2
            DO 46 J= I,N2
            B(I,J)= -HGGRY(I)*YGGRH(J)/YHY
      46 H(I,J)= H(I,J)+ B(I,J)
C
         YSIGMA= 0.
         DO 48 I= 1, N2
      48 YSIGMA= YSIGMA + SIGMA(I)*Y(I)
         DO 50 I= 1, N2
            DO 50 J= I, N2
         A(I,J)= SIGMA(I)*SIGMA(J)/ YSIGMA
            H(I,J)= H(I,J)+A(I,J)
      50 H(J,I)= H(I,J)
         GO TO 18
      51 PRINT 70
      52 CONTINUE
      70 FORMAT (/10X, 8HIANT=100)
         RETURN
         END
```

FTN5.48 02/04-69

```
      SUBROUTINE NEWRAPS
C
C     AUTHOR BERTIL LUNDGREN  MARCH 1969
C     MINIMIZATION OF FUNCTION OF N2 VARIABLES, STARTING AT X(I)
C     METHOD  NEWTON-RAPHSON
C     X(I) IS GIVEN THE COORDINATE OF THE MINIMUM
C     COMMON-ARRAY IS EXPLAINED IN SYSTEMID
C     INDATA  X(I) N2 METOD EPS1
C     OUTDATA  X(I)
C     REQUIRED SUBROUTINES  LINMINA LINMINB LINMINC VPRIMBIS
C
C
C
C
C
C
      DIMENSION UL(20,20)
      COMMON X(20), N2, M, STEGTID, METOD,  K, V, GRADV(20), G(20,20),
     F S(20), DELTA,  IANT, SGRADV, EPS1, YM(200),YB(200)
      K= 2
      N= N2/2
      N1= N + 1
      IANT= 1
   05 CALL VPRIMBIS
C
C     COMPUTE NEW DIRECTION S(I)
   10 CALL DECOM ( N2, G, UL, ISING, 20 )
      IF ( ISING ) 12, 15, 12
   12 PRINT 20
      GO TO 65
   15 CALL SOLVE ( N2, UL, GRADV, S, 20 )
      CALL IMPRUV ( N2, G, UL, GRADV, S, ISING, 20 )
      IF ( ISING ) 18, 25, 18
   18 PRINT 21
      DO 19 I= 1, N2
   19 PRINT 61, ( G(I,J), J= 1, N2 )
C
C     IF G IS SINGULAR THEN X(I) IS CHANGED
      DO 22 I= 1, N2
   22 X(I)= 1.02*X(I)
      GO TO 05
   25 DO 26 I= 1, N2
   26 S(I)= -S(I)
C
C     LINEAR MINIMIZING
      SGRADV= 0.
      DO 27 I= 1, N2
   27 SGRADV= SGRADV + S(I)*GRADV(I)
   33 GO TO ( 35, 38, 40 ), METOD
   35 CALL LINMINA
      GO TO 45
   38 CALL LINMINB
      GO TO 45
   40 CALL LINMINC
   45 PRINT 50, (X(I), I= 1, N)
      PRINT 51 , V, ( X(I), I= N1,N2 )
```

```
C       TEST IF COMPUTATION WILL BE FINISHED
        IANT= IANT + 1
        IF ( IANT -100 ) 53, 53, 60
   53   SKALS= 0.
        DO 132 I= 1, N2
  132   SKALS= SKALS + S(I)*S(I)
        EPSKVA= EPS1*EPS1
        IF (SKALS - EPSKVA) 65, 10, 10
   60   PRINT170
   20   FORMAT ( 9X, 11H1G SINGULAR )
   21   FORMAT ( 9X, 11H3G SINGULAR )
   50   FORMAT (/19X, 10F10.4 )
   51   FORMAT ( 9X, E10.3, 10F10.4 )
   61   FORMAT ( /19X, 10E10.3 )
  170   FORMAT (/10X, 8HIANT=100)
   65   RETURN
        END
```

TN5.4B 14/03-69

```
      SUBROUTINE LINMINA
C
C     REFERENCE COMPUTER-JOURNAL VOL.6 1963
C     AUTHOR BERTIL LUNDGREN  MARCH 1969
C     LINEAR MINIMIZATION OF FUNCTION V OF N2 VARIABLES,
C     STARTING AT X(I) IN DIRECTION S(I)
C     X(I) IS GIVEN THE COORDINATE OF THE LINEAR MINIMUM
C     COMMON-ARRAY IS EXPLAINED IN SYSTEMID
C     REQUIRED SUBROUTINE VPRIMBIS
C     INDATA  X(I) S(I) N2 SGRADV K
C     OUTDATA  X(I) GRADV(I) AND DEPENDING ON K G(I,J)
C
C
      COMMON X(20), N2, M, STEGTID, METOD,  K, V, GRADV(20), G(20,20),
     F S(20), DELTA,  IANT, SGRADV, EPS1, YM(200),YB(200)
      DIMENSION OLDX(20), Y(20)
      R= N2
      ETA= AMIN1( 1. ,   -R*V/SGRADV)
      IB= 0
   05 DO 10 I= 1, N2
   10 OLDX(I)= X(I)
      IB= IB + 1
      OLDV=V
      DO 20 I= 1,N2
   20 X(I)= X(I) + ETA*S(I)
      CALL VPRIMBIS
      GVYGGRS= 0.
      DO 22 I= 1, N2
   22 GVYGGRS= GVYGGRS + GRADV(I)*S(I)
      IF ( GVYGGRS ) 23, 40, 25
   23 ETA= 2.*ETA
      SGRADV= GVYGGRS
      IF ( IB .LT. 10 ) GO TO 05
C     THE MINIMUM IS PASSED
   25 Z= 3.*(OLDV - V )/ETA + SGRADV  + GVYGGRS
      PRINT 90, IB
      WKVADR=  ( Z*Z - SGRADV *GVYGGRS )
      IF (WKVADR) 26, 27,27
   26 WKVADR= -WKVADR
      PRINT 45
   27 W= SQRTF (WKVADR)
   28 ALFA= ETA*( 1. - (GVYGGRS + W - Z)/(GVYGGRS - SGRADV  + 2.*W) )
      ALFA100= ALFA*0.01
C     COMPUTE NEW X(I)
   29 DO 30 I= 1, N2
   30 X(I)= OLDX(I) + ALFA*S(I)
      CALL VPRIMBIS
      IF ( V - OLDV ) 40, 35, 35
   35 ALFA= ALFA/2.
       IF ( ALFA - ALFA100 )  40, 40, 29
   45 FORMAT (9X, 10HW IMAGINAR)
   90 FORMAT (  1X, 2HIB, I3  )
   40 RETURN
      END
```

FTN5.48 18/03-69

```
      SUBROUTINE LINMINB
C
C
C     AUTHOR BERTIL LUNDGREN  MARCH 1969
C     LINEAR MINIMIZATION OF FUNCTION V OF N2 VARIABLES,
C     STARTING AT X(I) IN DIRECTION S(I)
C     X(I) IS GIVEN THE COORDINATE OF THE LINEAR MINIMUM
C     COMMON-ARRAY IS EXPLAINED IN SYSTEMID
C     REQUIRED SUBROUTINE VPRIMBIS
C     INDATA  X(I) S(I) N2 K
C     OUTDATA  X(I) GRADV(I) AND DEPENDING ON K G(I,J)
C
      COMMON X(20), N2, M, STEGTID, METOD,  K, V, GRADV(20), G(20,20),
     F S(20), DELTA,  IANT, SGRADV, EPS1, YM(200),YB(200)
      DIMENSION OLDX(20)
      DO 05 I= 1,N2
   05 OLDX(I)= X(I)
      OLDV= V
      ALFA= 1.
      ALFAT= 0.0001*ALFA
C
   15 DO 20 I= 1, N2
   20 X(I)= OLDX(I) + ALFA*S(I)
      CALL VPRIMBIS
      IF (V - OLDV) 40, 30, 30
   30 ALFA= 0.5*ALFA
      IF ( ALFA - ALFAT ) 40,40,15
   40 RETURN
      END
```

FTN5.4B 02/04-69

```
      SUBROUTINE LINMINC
C
C     AUTHOR BERTIL LUNDGREN  MARCH 1969
C     LINEAR MINIMIZATION OF FUNCTION V OF N2 VARIABLES,
C     STARTING AT X(I) IN DIRECTION S(I)
C     X(I) IS GIVEN THE COORDINATE OF THE LINEAR MINIMUM
C     COMMON-ARRAY IS EXPLAINED IN SYSTEMID
C     INDATA  X(I) S(I) N2 SGRADV K
C     OUTDATA  X(I) GRADV(I) AND DEPENDING ON K G(I,J)
C
C     REQUIRED SUBROUTINE VPRIMBIS
C
C
C
      COMMON X(20), N2, M, STEGTID, METOD,  K, V, GRADV(20), G(20,20),
     F S(20), DELTA,  IANT, SGRADV, EPS1, YM(200),YB(200)
      DIMENSION XR(20), OLDX(20)
C
      KK= K
      K= 0
      IB= 0
      R= N2
      ETA= AMIN1( 1. ,   -R*V/SGRADV)
      OLDV= V
      DO 01 I= 1, N2
   01 OLDX(I)= X(I)
   02 DO 03 I= 1, N2
   03 X(I)= OLDX(I) + ETA*S(I)
      CALL VPRIMBIS
      V= 0.9*V
      IF ( V .LT. OLDV ) GO TO 04
      PRINT 90, IB
      ETA= 0.1*ETA
      GO TO 02
C
   04 DO 05 I= 1, N2
   05 X(I)= OLDX(I)
   06 IB= IB + 1
      IF ( IB .GT. 3 ) ETA= 2.*ETA
      IF ( IB .GT. 10 ) GO TO 70
      DO 10 I= 1, N2
   10 X(I)= X(I) + ETA*S(I)
      CALL VPRIMBIS
      IF ( V .LT. OLDV ) GO TO 06
C     THE MINIMUM IS PASSED
      ETA= 2.*ETA
      PRINT 90, IB
   90 FORMAT (  1X, 2HIB, I3  )
      DO 15 I= 1, N2
      XR(I)= X(I)
   15 X(I)= XR(I) - 0.382*ETA*S(I)
      CALL VPRIMBIS
      VR= V
      IH= 1
      Z= 0.618
      IT= 0
```

FTN5.48 02/04-09

```
      20 IT= IT + 1
         DO 25 I= 1, N2
      25 X(I)= XR(I) - Z*ETA*S(I)
         CALL VPRIMBIS
         GO TO (30, 35 ), IH
      30 VL= V
         GO TO 40
      35 VR= V
      40 IF ( VL-VR ) 45, 45, 55
      45 IH= 1
         VR= VL
         DO 50 I= 1, N2
      50 XR(I)= XR(I) - 0.382*ETA*S(I)
         ETA= 0.618*ETA
         Z= 0.618
         GO TO 60
      55 IH= 2
         VL= VR
         ETA= 0.618*ETA
         Z= 0.382
      60 IF ( IT .LE. 5 ) GO TO 20
         DO 65 I= 1, N2
      65 X(I)= XR(I)
      70 K=KK
         CALL VPRIMBIS
         RETURN
         END
 LOAD
***BINARY DECK***
 RUN,5, 1500, , ,
```

FTN5.4B 02/04-69

```
      SUBROUTINE VPRIMBIS
C
C
C     AUTHOR BERTIL LUNDGREN  MARCH 1969
C     COMPUTES LOSS-FUNCTION V, GRADIENT GRADV(I), AND
C     SECOND-DERIVATIVE G(I,J) FOR A DYNAMIC SYSTEM WITH PARAMETERS X(I)
C     COMMON-ARRAY IS EXPLAINED IN SYSTEMID
C     INDATA  X(I) N2 M STEGTID K YM(I)
C     OUTDATA  V GRADV(I) G(I,J) YB(I)
C     REQUIRED SUBROUTINE MEXP7T
C
C     COMPUTES GRADIENT WITH THEORY FROM DYNAMIC SYSTEMS   VERSION 2
C
C
C
C
C
      COMMON X(20), N2, M, STEGTID, METOD,  K, V, GRADV(20), G(20,20),
     F S(20), DELTA,  IANT, SGRADV, EPS1, YM(200),YB(200)
      DIMENSION ASTEG(20,20),DY(200),SLA(20)  ,XX(20),RES(20,20),
     F GRADYB(20), GYBS(20,100), C(20,20)
C
C
C
C     INITIAZING
      N= N2/2
      NM= N - 1
      DO 10 I= 1, N2
      DO 10 J= 1, N2
   10 ASTEG(I,J)= 0.
      DO 15 I= 1, NM
      NPI= N+I
      ASTEG(NPI+1,NPI)= STEGTID
   15 ASTEG(I+1,I)= STEGTID
      DO 20 I= 1, N
      NPI= N+I
      ASTEG(1,I)= -X(I)*STEGTID
      ASTEG(N+1,NPI)= -X(I)*STEGTID
   20 ASTEG(1,NPI)= -X(NPI)*STEGTID
      D= X(N+1) - YM(1)
      V= D*D
      YB(1)= X(N+1)
      DO 21 I= 1, N2
      GRADYB(I)= 0.
      GRADV (I)= 0.
   21 GYBS(I,1)= 0.
      GRADYB(N+1)= 1.
      GYBS(N+1,1)=GRADYB(N+1)
      GRADV(N+1)= GRADYB(N+1)*2.*D
      IF ( K ) 25, 25, 28
   25 CALL MEXP7T ( ASTEG, RES, N, 20, 1 )
      GO TO 29
   28 CALL MEXP7T ( ASTEG, RES, N2,20, 1 )
   29 DO 30 I= 1, N
   30 SLA(I)= 0.
      SLA(1)= 1.
```

```
C       START CALCULATE LOSSFUNCTION V
        DO 50 IS= 2, M
        DO 35 I= 1, N
        R= 0.
        DO 34 J= 1, N
   34   R= R + RES(I,J)*SLA(J)
   35   XX(I)= R
        DO 38 I= 1, N
   38   SLA(I)= XX(I)
        Y= 0.
        DO 40 I= 1, N
        NPI= N+I
   40   Y= Y + X(NPI)*XX(I)
        YB(IS)= Y
        D= Y - YM(IS)
        V= V + D*D
        DY(IS)= D
   50   CONTINUE
        IF ( K .EQ. 0 ) GO TO 90
C       START CALCULATE GRADIENT GRADV
        DO 80 IS= 2, M
        DO 55 I= 1, N2
   55   SLA(I)= GRADYB(I)
        DO 60 I= 1, N2
        R= 0.
        DO 58 J= 1, N2
   58   R= R + RES(I,J)*SLA(J)
   60   GRADYB(I)= R
        DO 65 I= 1, N2
   65   GRADV(I)= GRADV(I) + 2.*DY(IS)*GRADYB(I)
        GO TO ( 80, 70 ), K
   70   DO 75 I= 1, N2
   75   GYBS(I,IS)= GRADYB(I)
   80   CONTINUE
        IF ( K .EQ. 1 ) GO TO 90
C       START CALCULATE SECOND-DERIVATIVE
        DO 85 I= 1, N2
        DO 85 J= I, N2
        G(I,J)= 0.
        DO 85 IS= 1, M
        G(I,J)= G(I,J) + 2.*GYBS(I,IS)*GYBS(J,IS)
   85   G(J,I)= G(I,J)
   90   RETURN
        END
```

FTN5,48 18/03-69

```
      SUBROUTINE PLOT ( YM, YB, M, ISYMM, SKALFAK )
C
C     AUTHOR BERTIL LUNDGREN  MARCH 1969
C     PLOT FOR TWO FUNCTIONS YM(T) AND YB(T), USING A RASTER OF 56 LINES
C     M IS THE NUMBER OF MEASUREPOINTS
C     IF ISYMM=1 SKALFAK*YM(T) SHALL BE BETWEEN -25 AND +30 FOR ALL T
C     IF ISYMM=2 SKALFAK*YM(T) SHALL BE BETWEEN  -5 AND +50 FOR ALL T
C
      DIMENSION YM(100), YB(100), NRAD(56)
      INTEGER ZERO, SEVEN, TRE, BLANK, STAR
      DATA (ZERO=1H0),(SEVEN=1H7),(TRE=1H3),(BLANK=1H ),(STAR=1H*)
     F ,(II=1HI)
      GO TO ( 01, 05 ), ISYMM
   01 PRINT 60, SKALFAK
      PRINT 61
      PRINT 62
      GO TO 08
   05 PRINT 70, SKALFAK
      PRINT 71
      PRINT 72
   08 DO 45 I= 1,M
   15 GO TO ( 20, 25 ), ISYMM
   20 IYM= YM(I)*SKALFAK + 26.
      IYB= YB(I)*SKALFAK + 26.
      GO TO 27
   25 IYM= YM(I)*SKALFAK + 6.5
      IYB= YB(I)*SKALFAK + 6.5
   27 DO 30 J= 1, 56
   30 NRAD(J)= BLANK
      IF ( ISYMM .EQ. 1 ) NRAD(26)= II
      IF ( ISYMM .EQ. 2 ) NRAD(6)= II
   32 NRAD(IYM)= ZERO
      IF ( IYB ) 35, 36, 36
   35 NRAD(1)= STAR
      GO TO 42
   36 IF ( IYB-56 ) 40, 40, 38
   38 NRAD(56)= STAR
      GO TO 42
   40 NRAD(IYB)= SEVEN
      IF ( IYB .EQ. IYM ) NRAD(IYB)= TRE
   42 IF ( M .GT. 28 ) GO TO 43
      PRINT 50, YM(I), YB(I), NRAD
      GO TO 45
   43 PRINT 55, YM(I), YB(I), NRAD
   45 CONTINUE
C
   50 FORMAT ( /4X, 2(F10.5,6X), 56A1)
   55 FORMAT ( 4X, 2(F10.5, 6X), 56A1 )
   60 FORMAT ( 1H1,7X,4HYM 0,11X,4HYB 7, 34X,1H*,5X,10HSKALFAKTOR,F10.5)
   61 FORMAT ( 8X,4H****,11X,4H****,34X,1H* )
   62 FORMAT(36X,6H54321*,2(10H9876543210*),3(10H1234567890*) )
   70 FORMAT ( 1H1,7X,4HYM 0,11X,4HYB 7, 14X,1H*,5X,10HSKALFAKTOR,F10.5)
   71 FORMAT ( 8X,4H****,11X,4H****,14X,1H* )
   72 FORMAT ( 36X,6H54321*, 5(10H1234567890*) )
   80 RETURN
      END
```

7. Förslag till förbättring av programmet

I programmet finns inget test på att det beräknade systemet är stabilt. Detta kan gäras ganska enkelt utan att beräkna systemets poler.

Isamband med plottningen beräknar man utsignalen YB och förlustfunktionen V för fler mätpunkter än de som använts vid minimeringen. Om V då blir avsevärt större (t ex mer än tredubblas) så anser man att systemet är instabilt. Förutsättningen är att $YM_t = 0$ för de efterföljande tidpunkterna.

Ändringen i PROGRAM SYSTEMID kan då bli följande fr o m sats 62 :

```
62 VTRE= 3.*V
   MM= M+1
   M= M+M/2
   IF ( M .GT. 200 ) M= 200
   DO 63 I= MM, M
63 YM(I)= 0.
   K= 0
   CALL VPRIMBIS
   CALL PLOT ( YM, YB, M, ISYMM, SKALFAK )
   IF ( V .LT. VTRE    ) GO TO 64
   PRINT 103
103 FORMAT ( //18HSYSTEMET INSTABILT )
   GO TO 98
 64 M= MM-1
   IF ( V-SLUTV ) 80, 80, 65

   osv fr o m sats 65
```

KAP 6. KÖRNINGAR PÅ DATAMASKIN

1. Testexempel, Rosenbrocks banan

För att testa subrutinerna FLEPO, NEWRAPS och de linjära minimeringsmetoderna har jag minimerat en potensfunktion

$$V(x_1,x_2)= 100(x_2-x_1^2)^2+ (1-x_1)^2$$

Funktionen kallas Rosenbrocks banan och har även använts i REF 2. Utseendet är en mycket brant dal längs kurvan $x_2=x_1^2$ med en minimipunkt i (1,1).

Funktionens branthet åskådliggöres genom att jämföra $V(0,0)=1$ och $V(0,1)=101$.

Som startpunkt valdes (-1.2, 1.0). FLEPO med LINMINA hittade minimet efter 21 iterationer, med LINMINB konvergerade beräkningen inte. NEWRAPS med LINMINA hittade minimet efter 17 iterationer och med LINMINB efter 19 iterationer. LINMINC var inte skriven då försöken utfördes. I LINMINA utfördes ingen stegning tills minimet var passerat.

Försöket styrker påståendet i kap 2 att Fletcher-Powells metod kväver en noggrann linjär minimering, dvs LINMINB duger inte. LINMINB duger däremot utmärkt till Newton-Raphsons metod, bara två iterationer mer än med LINMINA.

Räknetiderna var ungefär samma, omkring en halv sekund. Eftersom VPRIMBIS i princip bara anropas en gång från LINMINB, kommer räknetiden med den metoden att vara fördelaktigare ju mera komplicerad förlustfunktionsberäkningen blir.

2. Första ordningens system

För att testa VPRIMBIS i Version 1 identifierade jag systemet $y(t)=100e^{-0.5t}$, dvs $a_1=0.5$, $b_1=100$.
Jag valde snälla startvärden $a_1=0.55$, $b_1=115$.

Mätpunkterna var: 100.00, 60.65, 36.79, 22.31, 1353, 8.20
4.98, 3.02 , med 1 sek mellan vardera.

FLEPO LINMINA		NEWRAPS LINMINB	
a_1	b_1	a_1	b_1
0.550	115.00	0.550	115.00
0.587	114.99	0.503	99.93
0.509	99.93	0.500	100.00
0.499	99.97		
0.500	100.00		

NEWRAPS hittar minimet approximativt redan i första iterationen, medan FLEPO först går i brantaste riktningen, som tydligen inte är den bästa.

3. Andra ordningens system

Med VPRIMBIS i Version 1 identifierades samma system, som man använt i REF 1.

Överföringsfunktionen är $G(s)= \frac{200}{s^2+s+1}$

dvs $a_1=1$, $a_2=1$, $b_1=0$, $b_2=200$

Startvärden: $a_1=0.9$, $a_2=1.1$, $b_1=5$, $b_2=220$

Utsignalens mätvärden: 0.0 106.80 83.80 26.70 -9.85
-17.60 -10.25 -1.54

med samplingstiden 1 sek.

FLEPO LINMINA					NEWRAPS LINMINB				
V	a_1	a_2	b_1	b_2	a_1	a_2	b_1	b_2	V
	0.900	1.100	5.00	220.00	0.900	1.100	5.00	220.00	
86.0	1.080	1.015	4.99	219.99	1.011	1.024	-0.18	203.43	6.0
71.0	1.098	1.041	4.99	219.99	1.002	0.999	0.02	200.19	0.02
29.0	1.089	1.049	-1.23	217.98	1.001	0.999	0.01	200.15	0.01
5.0	1.039	1.004	-0.37	203.72					
0.040	0.999	0.998	0.06	199.93					
0.018	1.001	1.000	0.02	200.18					

Räknetid 7sek Räknetid 2 sek

Med VPRIMBIS i Version 2 erhölls följande räknetider (sek) för samma system

	LINMINA	LINMINB	LINMINC
FLEPO	2.1		4.0
NEWRAPS	0.9	2.0	

4. Signalanpassade filter

Från institutionen för Teletransmissionsteori har hämtats två exempel. Man vill beräkna en approximativ överförings-funktion till två signalanpassade filter. Närmare upp-lysning om bakbrunden till problemet lämnas i REF 1.

Kontentan av problemet är att man vill beräkna en över-föringsfunktion av så låg ordning som möjligt, vars impulssvar följer ett givet sådant inom vissa tolerans-gränser. Se fig 6.1 och fig 6.2.

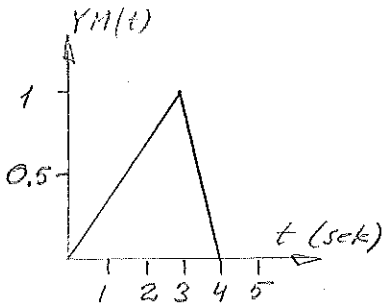


Fig 6.1 Impulssvar 1

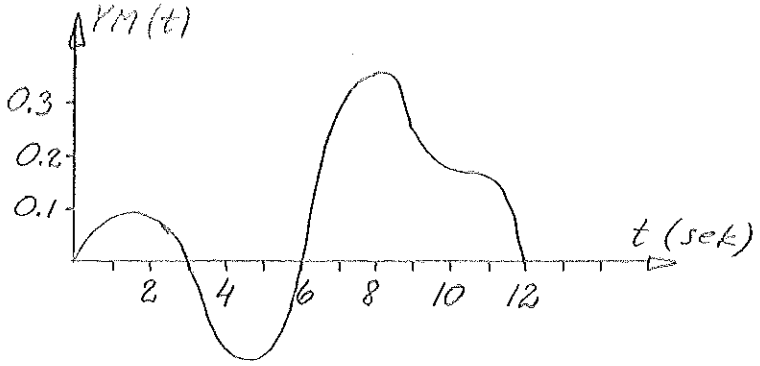


Fig 6.2 Impulssvar 2

Impulssvar 1 skall följa triangeln så väl som möjligt, och sedan vara så nära noll som möjligt. På motsvarande sätt för Impulssvar 2.

Körningar med Impulssvar 1

Impulssvar 1 kördes med VPRIMBIS i Version 1 och LINMINA utan inbyggd stegning. FLEPO konvergerade inte, så jag förbättrade LINMINA med stegning, se kap 4. Då erhölls följande iterationsantal och räknetider (sek) med 16 mät-punkter.

N	Tid/iter	Ant iter
2	0.7	7
3	4.0	14
4	15.7	Ej färdig

Den långa räknetiden för N=4 beror antagligen på att ett stort antal stegningar har skett i LINMINA. Jag beslöt då att fördubbla ETA för varje stegning och att begränsa stegningsantalet till 10.

Med VPRIMBIS i Version 2 erhölls följande resultat.
Startvärden a_1=0.10 a_2=0.25 b_1=0.0 b_2= 0.40

	N	2	3	4	5	6	6
FLEPO LINMINA	Ant iter	8	18	25	19	50	44
	Räknetid(sek)	2.4	13	35	47	240	190
	M	16	16	16	16	16	32
	V	0.496	0.223	0.100	0.074	0.034	0.050
FLEPO LINMINO	Ant iter	8	23	50	50	31	13
	Räknetid(sek)	4.0	20	75	100	100	44
	M	20	20	20	20	20	40
	V	0.520	0.240	0.150	0.100	0.086	0.128
NEWRAPS LINMINB	Ant iter	12	34	34	39		
	Räknetid(sek)	2.8	47	29	69		
	M	40	40	40	40		
	V	0.894	0.381	0.162	0.069		
	G singulär	nej	ja	nej	ja		

Ur körningarnas resultat kan man dra följande slutsatser, som gäller identifiering av Impulssvar 1 :

1. Iterationsgränsen 50 var för liten. Den utökades därför till 100.
2. LINMINA räknar snabbare än LINMINC för låga ordningstal (lägre än 6).
3. Det finns ingen markant skillnad i räknetiden, då man jämför NEWRAPS och FLEPO.
4. Andraderivatmatrisen G blev singulär för N=3 och N=5 men minimeringen fortsatte normalt.

I programmet finns ingen kontroll av att man beräknar ett stabilt system. Med FLEPO beräknades ett instabilt system. Detta kan bero på att minimeringen inte fick fullföljas för varje ordningstal på grund av iterationsgränsen. Om inte iterationsgränsen var boven i dramat, eliminerar man problemet genom att ta med flera nollor efter triangeln.

I figurerna 6.3, 6.4, 6.5 och 6.6 återges resultatet av körningen med NEWRAPS, LINMINB och 40 mätpunkter.

N = 2 V = 0.894

$$G(s) = \frac{-0.2259s+0.8708}{s^2+0.7185s+0.4886}$$

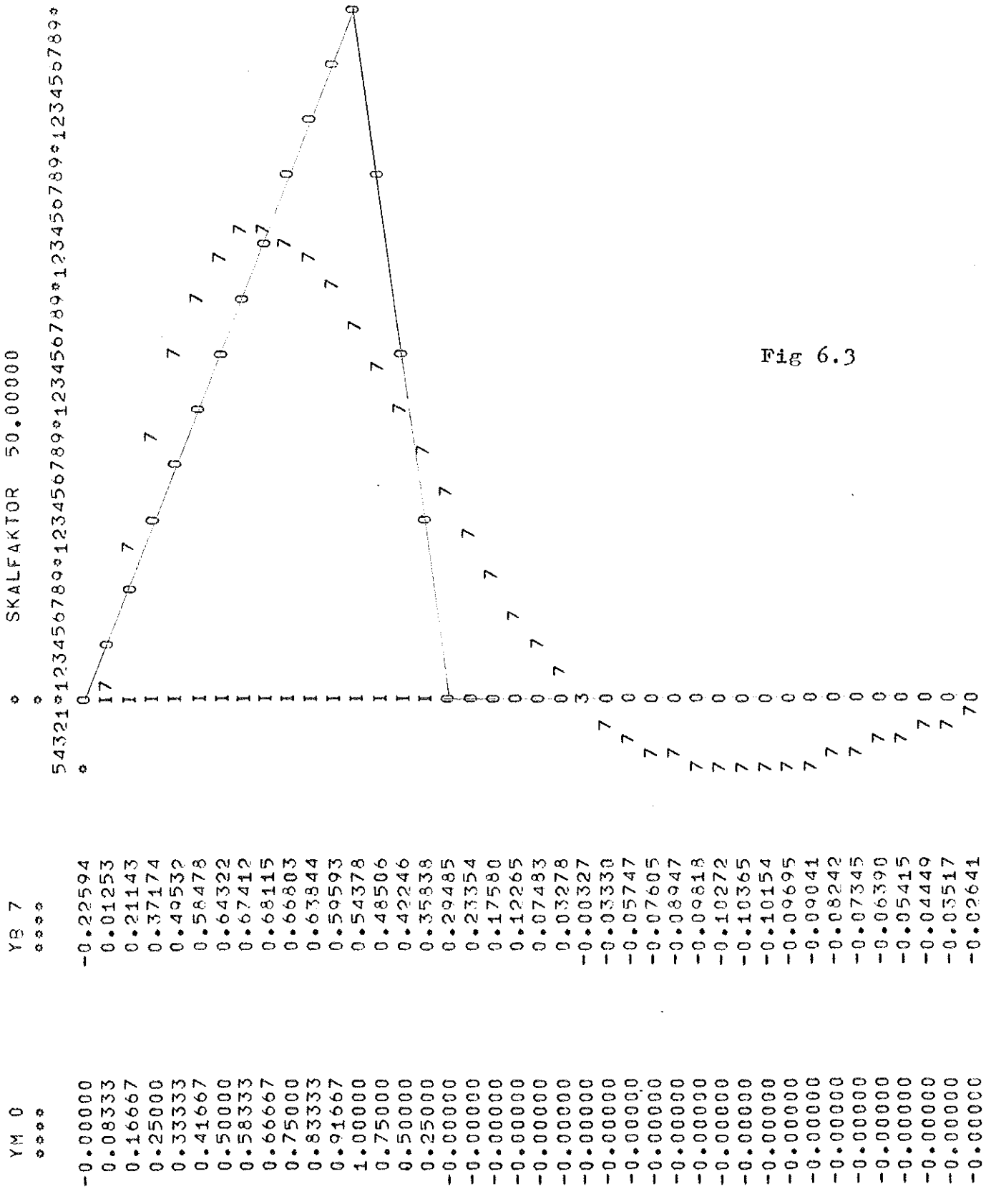


Fig 6.3

N = 3 V = 0.381

$$G(s) = \frac{0.1247s^2-0.4496s+1.6891}{s^3+1.3360s^2+1.9569s+0.8126}$$

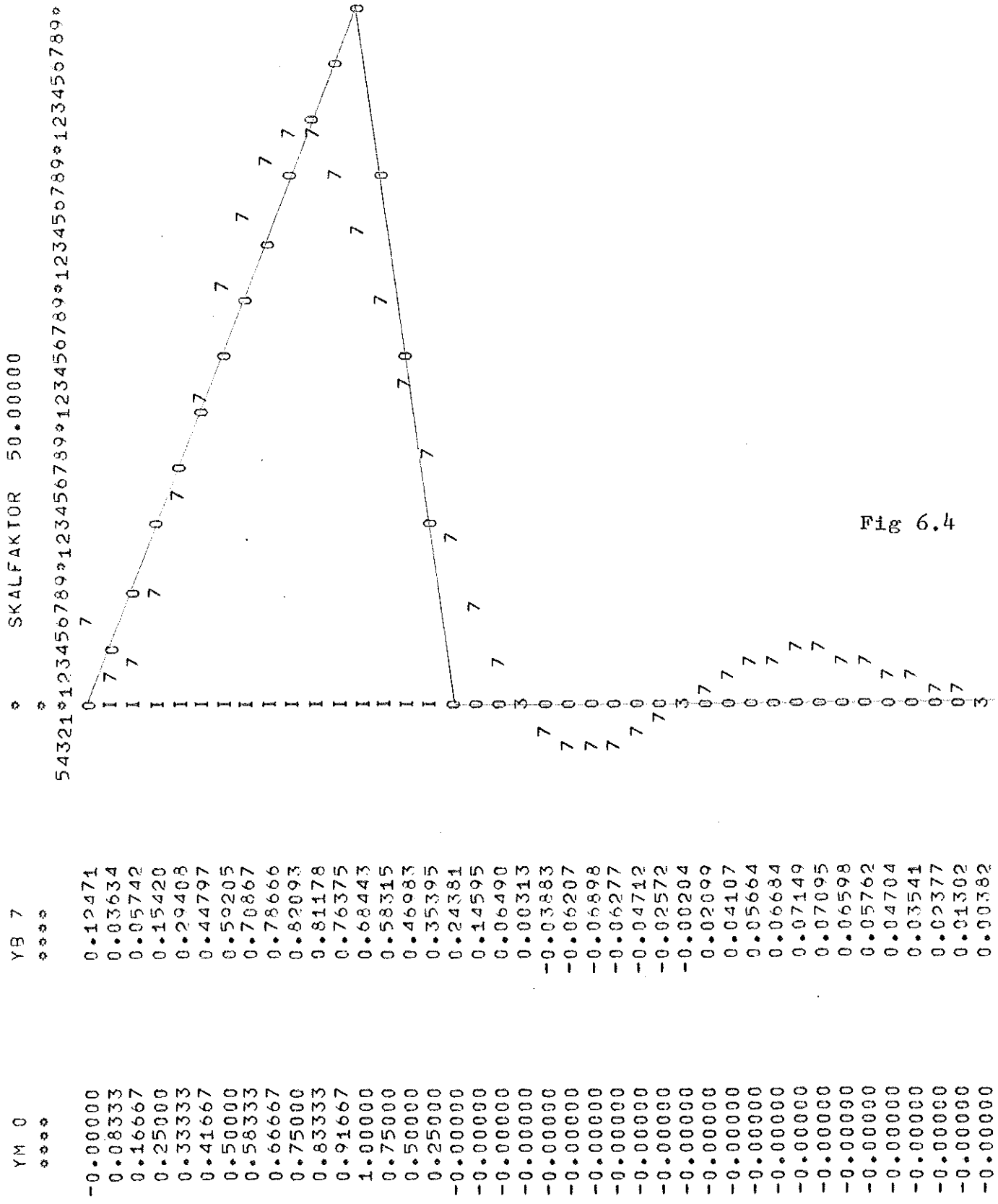


Fig 6.4

N = 4 V = 0.162

$$G(s) = \frac{-0.0637s^3+1.1891s^2-1.7244s+5.5205}{s^4+2.0671s^3+5.4213s^2+5.2036s+2.8279}$$

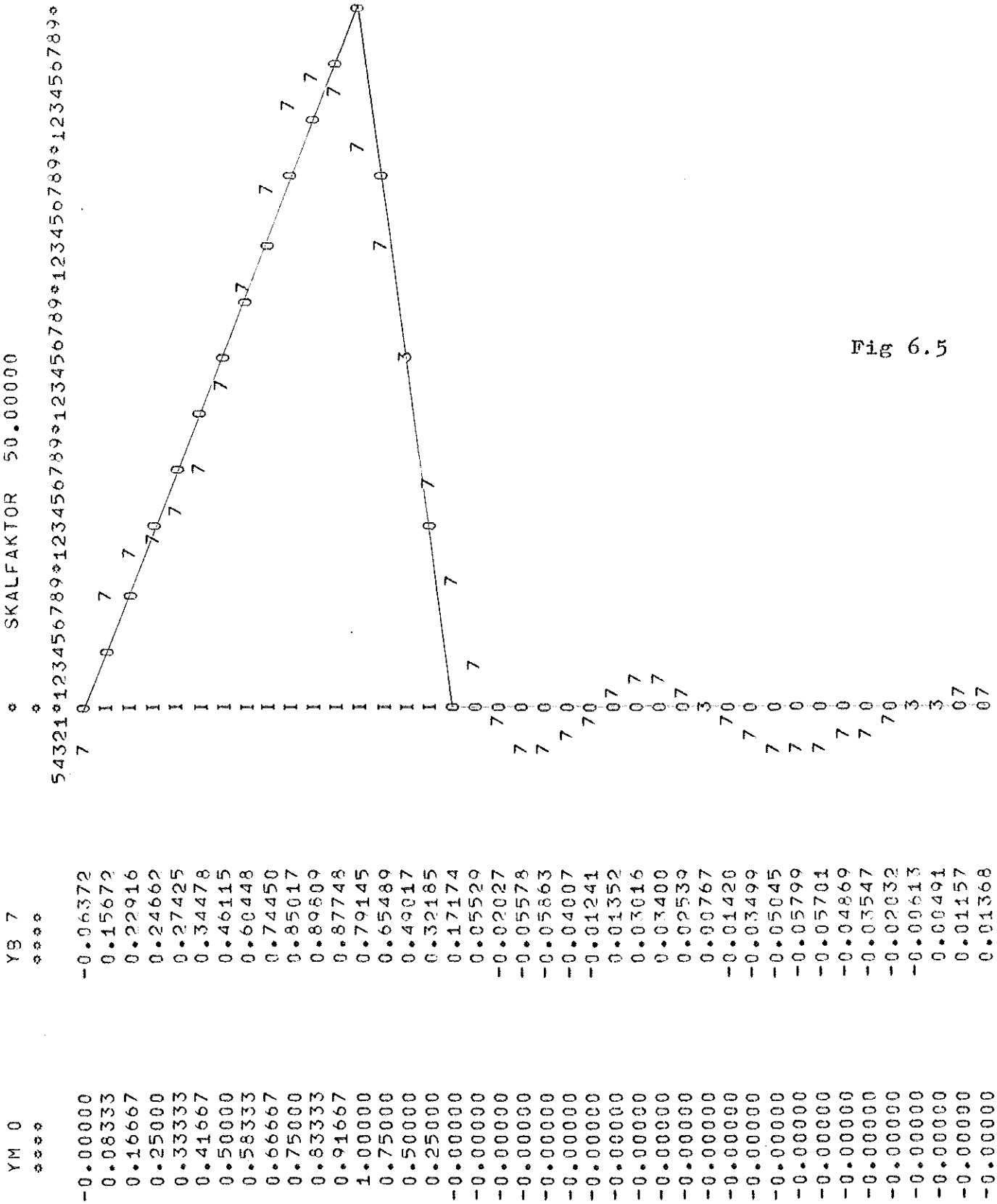


Fig 6.5

N = 5 V = 0.069

$$G(s) = \frac{0.0299s^4 - 0.5419s^3 + 4.7188s^2 - 6.8051s + 20.7246}{s^5 + 2.9354s^4 + 11.5065s^3 + 18.2088s^2 + 21.5901s + 10.2246}$$

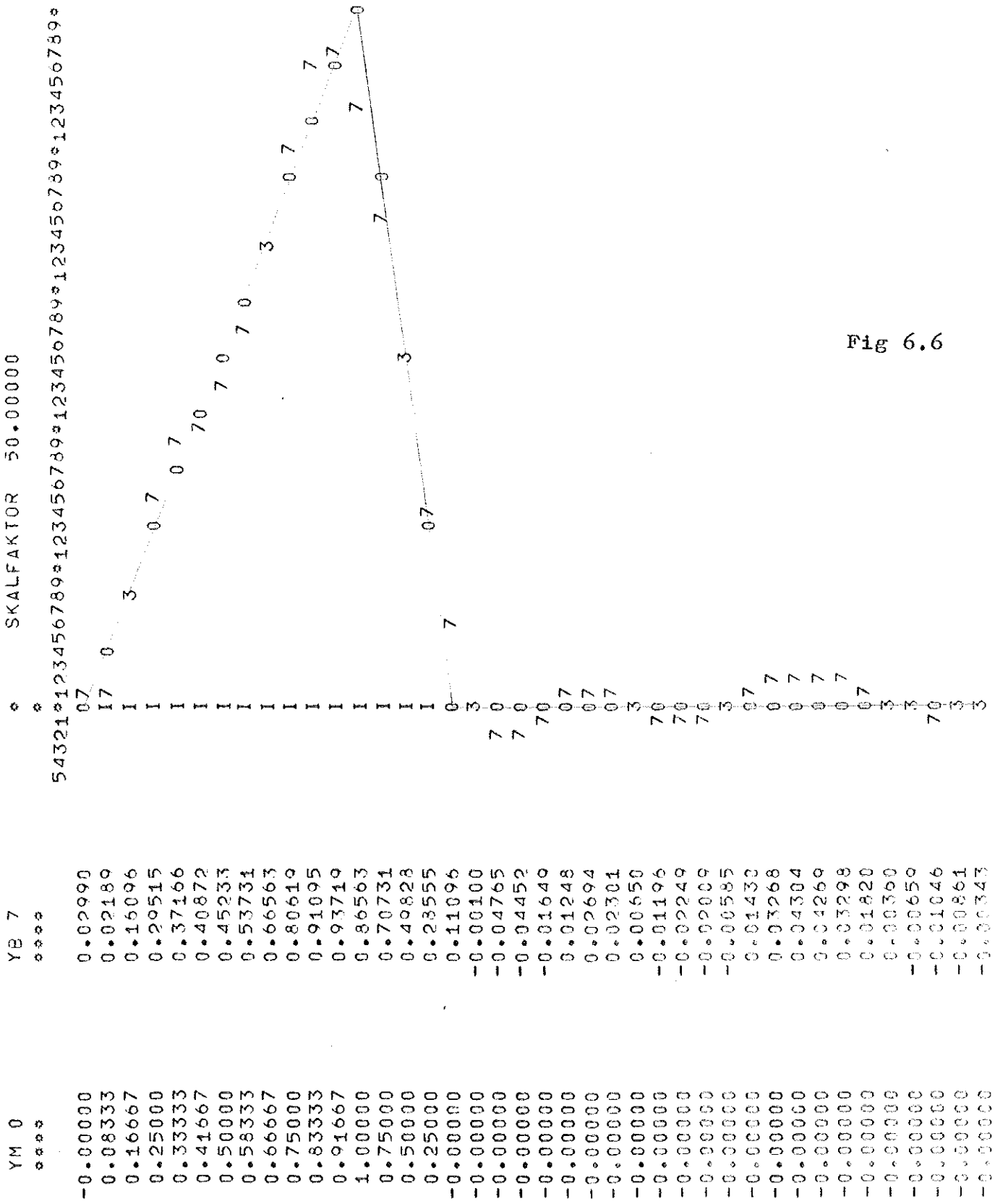


Fig 6.6

Körningar med Impulssvar_2_

Startvärden: $a_1=0.12$ $a_2=0.75$ $b_1=0.10$ $b_2=0.12$

Resultat:

	N	2	3	4	4	5
FLERP LINMINC	Ant iter	11	8	17	9	28
	Räknetid(sek)	8	8	27	17	85
	M	28	28	28	56	56
	V	0.279	0.154	0.037	0.071	0.027
NEWRAPS LINMINB	Ant iter	8	12	12		17
	Räknetid(sek)	3	6	19		18
	M	56	56	56		56
	V	0.547	0.292	0.071		0.027
	G singulär	nej	ja	ja		nej

Resultatet för N=5 återges i fig 6.7

N = 5 V = 0.027

$$G(s) = \frac{0.0113s^4 - 0.1555s^3 + 0.4633s^2 - 0.3044s + 0.0899}{s^5 + 1.3937s^4 + 1.9904s^3 + 1.1855s^2 + 0.5198s + 0.0848}$$

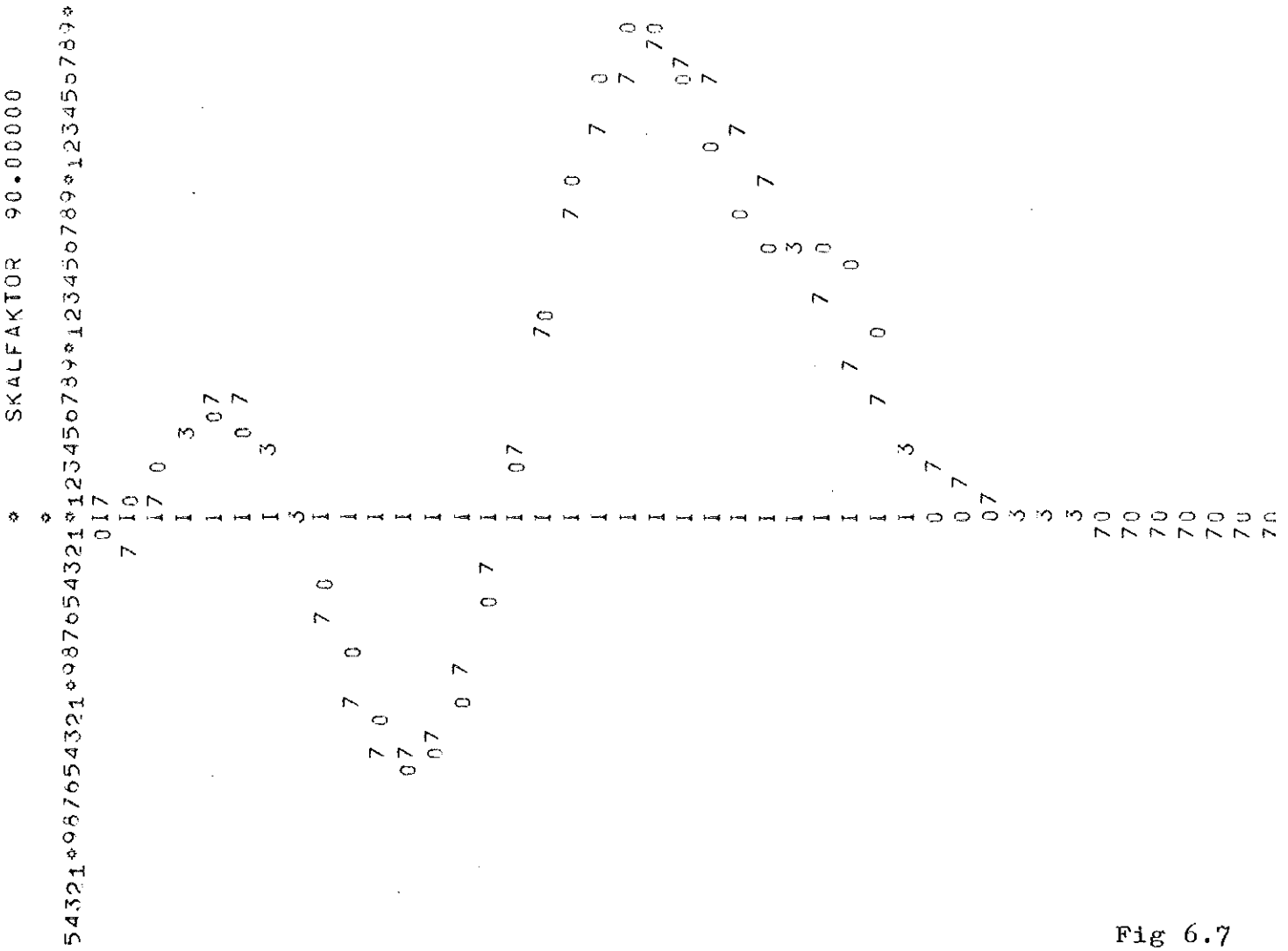


Fig 6.7