IDENTIFIERING AV LINEÄRA SYSTEM

MED HJÄLP AV IMPULSSVAR

GÖSTA JOHNSSON o ÖRJAN ROSELL

Rapport RE - 31 juni 1968

Identifiering av lineära, tidsinvarianta deterministiska system med hjälp av pulsanalys.

Vid institutionen för regleringsteknik vid tekniska högskolan i Lund har som examensarbete undersökts möjligheten att med hjälp av endast impulssvaret från ett okänt system bestämma överföringsfunktionen. Det kontinuerliga systemet samplas och kan genom lämplig transformation i tillståndsrummet beskrivas av 2n variabler, $(a_1 \ldots a_n, b_1 \ldots b_n)$, där n är systemets ordningstal. Utsignalen $z_1(t)$ från denna matematiska representation anpassas enligt minsta kvadratmetoden till det i samplingspunkterna uppmätta impulssvaret ymätt(t). Förlustfunktionen blir $V = \sum_{t=0}^{N-1} (z_1(t) - \text{ymätt}(t))^2$ där N är antalet mätpunkter. $z_1(t)$ och också V är en funktion av de 2n variablerna $(a_1 \ldots a_n, b_1 \ldots b_n)$. Man leds alltså till att minimera en funktion av flera variabler, vilket är möjligt med ett flertal olika numeriska metoder. I de två metoder som här har använts behöver gradienten av V, $|g^k\rangle$, beräknas i varje iterationspunkt. Iterationsformel:

$|x^{k+1}\rangle = |x^k\rangle - \alpha^k P^k |g^k\rangle$ där $\langle x^k| = (a_1^k \ldots a_n^k, b_1^k \ldots b_n^k)$

I Newton-Raphsons metod är 2n x 2n matrisen P^k identisk med andraderivatmatrisens invers (G^{-1}). (Referens: Saaty, Brown: Nonlinear Mathematics)
I Fletcher-Powells metod är P^k en positivt definit symmetrisk matris H. (Computer Journal vol.6 1963). α^k är en skalfaktor. Gradienten och andraderivatmatrisen av V låter sig beräknas ur systemekvationerna. Ett enkelt 2:a ordningens system med överföringsfunktionen $G(s) = \frac{1}{s^2+s+1}$ kopplades upp på analogimaskin och impulssvaret uppmättes. Förlustfunktionen V minimerades för n=1,2 och 3. Programmen körda på SMIL i Lund. Mellan ordningstalen 1 och 2 fås en kraftig nedgång i V_{min}, vilket är väntat då systemet är av ordning 2. Det samplade systemet, som ges av koefficienterna i minimipunkten överföres till kontinuerligt system, varefter överföringsfunktionen beräknas till

$$G(s) = \frac{0.0004s + 1.001}{s^2 + 1.002s + 0.999}$$

Maximala felet i systemparametrarna är således 0.2%

För n=1 och 2 visar sig Newton-Raphsons metod vara snabbare än Fletcher-Powells. Startar man långt från minimipunkten blir minimeringstiden densamma med de båda metoderna. Med fler mätpunkter och system av högre ordning (n=3,4,5...) är Fletcher-Powells metod klart överlägsen då andraderivatmatrisens invers inte behöver beräknas, utan H-matrisen byggs upp med hjälp av endast gradienten. Mot slutet av minimeringen går H-matrisen mot andraderivatmatrisens invers G^{-1}.
Vid signaldetektering med hjälp av signalanpassade filter behövs system med visst impulssvar. Här undersöktes system med impulssvaret

och för ordningstalen n=3,4, och 5 erhölls, helt naturligt, en allt bättre anpassning till $z_1(t)$. För speciellt n=5 blev förhållandet (max.amplitud för t>4)/(max.amplitud för hela signalen) = 0.05

En praktisk svårighet uppstår då en puls skall sändas in. Här har förutsatts en diracpuls, och impulssvaret har simulerats på analogimaskin. Med lämpligt vald puls bör emellertid denna metod vara användbar för identifiering av lineära system.

Identifiering av lineära system med hjälp av impulssvar.

(Funktionsminimering med hjälp av Fletcher-Powells och Newton-Raphsons metoder)

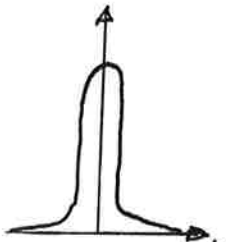
Gösta Johnsson

Örjan Rosell

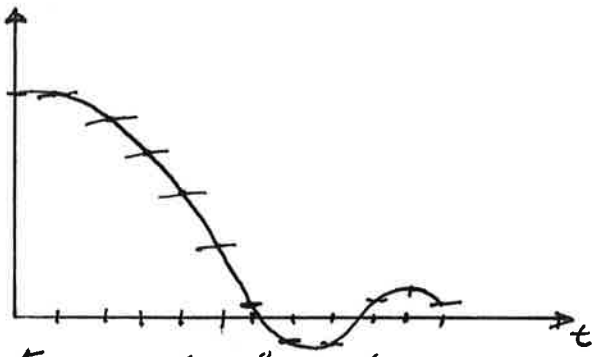
Innehåll.

Kapitel 1 Inledning.

Givet ett lineärt deterministiskt system med en ingång och en utgång. Pulssvaret ut uppmätes i diskreta tidpunkter (se figuren).



insignal



Exempel på pulssvar

Med hjälp av de erhållna värdena, y mätt(t) för t=0,1,2,..,N, skall systemparametrarna bestämmas.

Systemet kan beskrivas av:

$$\begin{cases} \dfrac{dx(t)}{dt} = A'x(t) + B'u(t) \\ x(0) = 0 \\ u(t) = \delta(t) \end{cases} \qquad (1.1)$$

där vi kan välja tillståndsrummet så att

$$A' = \begin{bmatrix} -a_1' & 1 & 0 & \cdots & 0 \\ -a_2' & 0 & 1 & & \\ \vdots & & & & \\ & & & & 1 \\ -a_n' & 0 & 0 & \cdots & 0 \end{bmatrix} \quad \text{och} \quad B' = \begin{bmatrix} b_1' \\ b_2' \\ \vdots \\ b_n' \end{bmatrix}$$

Således ett system på normalform med 2n parametrar.

Integrera 1.1 över tiden för pulsen in

$$x(0+) - x(0-) = B' \int_{0-}^{0+} \delta(t)\,dt = B'$$

Då x(0-)=0 fås systemet

$$\begin{cases} \dfrac{dx(t)}{dt} = A' x(t) \\ x(0+) = B' \end{cases}$$

När systemet samplas fås

$$\begin{cases} x(t+h) = e^{A'h} x(t) = \Phi' x(t) \\ x(0) = B' \\ y(t) = [1\,0\ldots0]\, x(t) \end{cases}$$

Gör transformationen z=Tx

$$\begin{cases} z(t+h) = T e^{A'h} T^{-1} z(t) \\ z(0) = T B' \\ y(t) = [1\,0\,0\ldots0]\, T^{-1} z(t) \end{cases}$$

Välj T så att

$$\begin{cases} z(t+h) = \begin{bmatrix} -a_1 & 1 & 0 & \cdots & 0 \\ -a_2 & 0 & 1 & \cdots & 0 \\ \vdots & & & & \\ -a_n & 0 & \cdots & \cdots & 0 \end{bmatrix} z(t) = \Phi z(t) \\ z(0) = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = B \\ y(t) = z_1(t) \end{cases} \qquad \ldots\ldots(1.2)$$

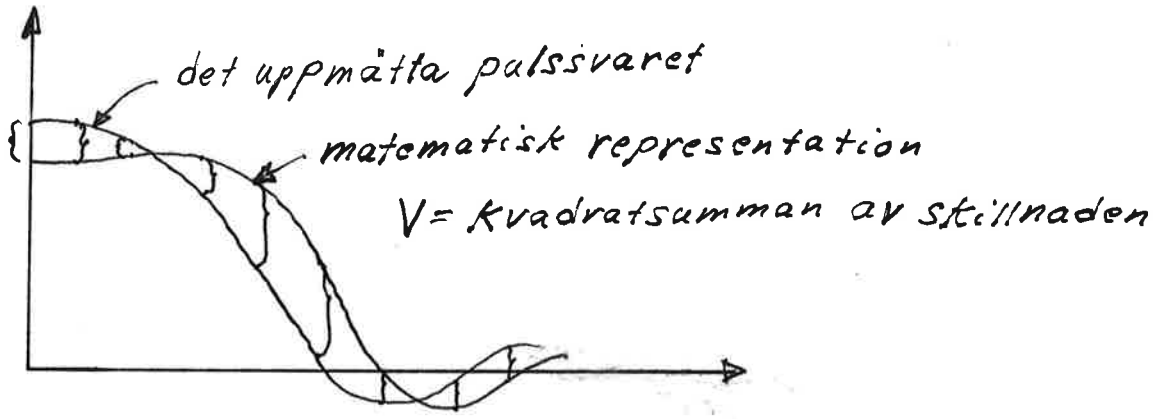
Vi bildar nu en förlustfunktion, denna väljes till

$$V = \sum_{t=0}^{N-1} (z_1(t) - y_{mätt}(t))^2 = V(a_1, \ldots a_n, b_1, \ldots b_n) = \qquad \ldots.(1.3)$$

$$= V(a, b) = V(\langle x \rangle)$$

Det gäller nu att till den uppmätta kurvan anpassa utsignalen, $y(t) = z_1(t)$, från vårt matematiska system så bra som möjligt, d.v.s. att minimera V, som är en funktion

av 2n variabler. Då detta uppnåtts, har vi, så nogrannt som möjligt, bestämt de 2n parametrar, som matematiskt beskriver det uppmätta systemet. Se figuren nedan.



För att numeriskt minimera en funktion av flera variabler finns ett flertal olika metoder. Här skall presenteras två, givna av Newton-Raphson och Fletcher-Powell.

Kapitel 2 Minimeringsmetoder

För minimeringen av vår förlustfunktion V, söker vi nu en iterativ metod, som utnyttjar att de partiella derivatorna är noll i minimet.

Det förutsättes alltså att den funktion, som skall minimeras är differentierbar. Sätt nu vår startpunkt vid iterationens början approximativt till radvektorn $(x_1, \ldots, x_n) = \langle x|$, minimet till $\langle x_0|$ och $\langle s| = \langle x_0 - x|$. Vi har då bl.a. $\langle x_0| = \langle x + s|$.
Taylorutveckling av de partiella derivatorna (gradienten) ger i minimet för funktionen $F(|x\rangle)$:

$$\left[\frac{\partial}{\partial x_i} F(|x\rangle)\right]_{|x_0\rangle} = \left[\frac{\partial}{\partial x_i} F(|x\rangle)\right]_{|x\rangle} + \sum_{j=1}^{n} \left[\frac{\partial^2}{\partial x_i \partial x_j} F(|x\rangle)\right]_{|x\rangle} \cdot s_j + O(s^2) = 0$$

för i=1,2,....n
Vi betecknar nu:

$$g_i = \left[\frac{\partial}{\partial x_i} F(|x\rangle)\right]_{|x\rangle}$$

$$G_{ij} = \left[\frac{\partial^2}{\partial x_i \partial x_j} F(|x\rangle)\right]_{|x\rangle}$$

Vi får med dessa beteckningar:

$$s = -G^{-1}[g + O(s^2)]$$

som ger

$$|x^{k+1}\rangle = |x^k\rangle - G^{-1}|g^k\rangle \qquad \ldots\ldots\ldots\ldots \text{(ekv. nr:2.1)}$$

där k betecknar iterationsnummer.
Vi inför nu en skalningsfaktor α^k på korrektionen ($G^{-1}|g^k\rangle$), för att undvika att en del olägenheter inträffar. (Se kap. 4)

Vi får så en iterationsformel.

$$|x^{k+1}\rangle = |x^k\rangle - \alpha^k G^{-1}|g^k\rangle \qquad \ldots\ldots\ldots\ldots (2.2)$$

Denna formel användes i Newton-Raphsons metod. I Fletcher-Powells metod bildas en positivt definit matris H med hjälp

av de partiella derivatorna i punkten $|x^k\rangle$ (se kap 5). H insättes istället för G^{-1} i formeln, d.v.s. vi får

$$|x^{k+1}\rangle = |x^k\rangle - \alpha^k H |g^k\rangle \qquad \ldots\ldots\ldots (2.3)$$

Allmännt kan vi skriva vår iterationsformel

$$|x^{k+1}\rangle = |x^k\rangle - \alpha^k \rho^k |g^k\rangle$$

Steglängden α^k väljes så att vår förlustfunktion V, får minimum längs linjen

d.v.s. en lineär minimering. För att utföra denna minimering, letar vi upp två punkter, med skilda α -värden, efter linjen. Dessa skall ha lägre förlustfunktion (V) än utgångspunkten, vilken motsvarar $\alpha = 0$. Vi får således tre punkter, till vilka vi kan anpassa en parabel. Ur parabelns minimipunkt får vi så vårt α^k. Detta utföres i algol-programmet huvudsakligen inom procedure M (se kap 3).

För att genomföra iterationen behöver vi enligt föregående bl.a. beräkna de partiella derivatorna(gradienten) och andraderivatorna (allt i utgångspunkten $|x^k\rangle$). Detta sker i procedure M (se kap 3)

Vi kan nu skissera iterationsförfarandet:

1) Start: $\langle x^0| = (a^0, b^0) = (a_1^0, \ldots . a_n^0, b_1^0, \ldots . . . b_n^0)$
   Val av startpunkt se kap. 6.
2) Beräkna i punkten $|x^k\rangle$:
   förlustfunktionen V, gradienten $|g^k\rangle$
   $$(\langle g^k| = (\frac{\partial V}{\partial a_1}, \ldots \frac{\partial V}{\partial a_n}, \frac{\partial V}{\partial b_1}, \ldots \frac{\partial V}{\partial b_n}))$$
   samt ρ^k där ρ^k är:
   a) En positivt definit matris H i Fletcher-Powells metod se kap. 5.
   b) G^{-1} i Newton-Raphsons metod. Andraderivatmatrisen (G) beräknas i procedure M (se kap 3) och dess invers se kap. 4.
3) α^k bestämmes genom vår lineära minimering (se kap.3).
4) Beräkna en ny punkt enligt:
   $$|x^{k+1}\rangle = |x^k\rangle - \alpha^k \rho^k |g^k\rangle$$
   Den nya punkten användes i 2) och förfarandet upprepas tills korrektionen $\delta^k = -\alpha^k \rho^k |g^k\rangle$ och $\delta^k = -\rho^k |g^k\rangle$ är tillräckligt små.

Algolprogrammen för iterationen presenteras för Newton-Raphsons metod i kap.4, för Fletcher-Powells i kap.5 .

Kap. 3 Procedure M

I båda minimeringsmetoderna behövs gradienten i varje iterationspunkt, dessutom behövs andraderivatmatrisen vid Newton-Raphsons metod. Gradienten och andraderivatmatrisen beräknas i en för båda programmen gemensam procedur procedure M. I detta kapitel beskrivs den matematiska bakgrunden samt beräkningsgången i algolprogrammet för procedure M(a, b, V, VA, VB, W, k).

3.a. Beräkning av gradienten

Vi har enligt 1.3 att

$$V = \sum_{t=0}^{N-1} (z_1(t) - y_{mätt}(t))^2$$

Derivering m.a.p. a_j ger

$$\frac{\partial V}{\partial a_j} = 2 \sum_{t=0}^{N-1} (z_1(t) - y_{mätt}(t)) \frac{\partial z_1(t)}{\partial a_j} \qquad \text{.....(3.1)}$$

Vi behöver alltså $\frac{\partial z_1(t)}{\partial a_j}$ och $\frac{\partial z_1(t)}{\partial b_j}$

Enligt 1.2

$$\begin{cases} z(t+h) = \begin{bmatrix} -a_1 & 1 & 0 & 0 \\ -a_2 & 0 & 1 & 0 \\ \vdots & & & \\ -a_n & 0 & \cdots & 0 \end{bmatrix} z(t) = \Phi z(t) \\ z(0) = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \end{cases}$$

Derivering m.a.p. $a_1 a_2 \ldots\ldots a_n$ med h=1

$$\begin{cases} \frac{\partial z}{\partial a_1}(t+1) = \Phi \frac{\partial z}{\partial a_1}(t) - \begin{bmatrix} z_1(t) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ \frac{\partial z}{\partial a_2}(t+1) = \Phi \frac{\partial z}{\partial a_2}(t) - \begin{bmatrix} 0 \\ z_1(t) \\ \vdots \\ 0 \end{bmatrix} \\ \vdots \\ \frac{\partial z}{\partial a_n}(t+1) = \Phi \frac{\partial z}{\partial a_n}(t) - \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ z_1(t) \end{bmatrix} \end{cases} \qquad \text{.....(3.2)}$$

Således gäller med $\xi^{-1} y(t) = y(t-1)$;

$$\begin{cases} \dfrac{\partial z_1}{\partial a_1}(t) = \dfrac{-\xi^{-1}}{1+a_1\xi^{-1}+a_2\xi^{-2}+\ldots+a_n\xi^{-n}} \cdot z_1(t) \\ \dfrac{\partial z_1}{\partial a_2}(t) = \dfrac{-\xi^{-2}}{1+a_1\xi^{-1}+a_2\xi^{-2}+\ldots+a_n\xi^{-n}} \cdot z_1(t) \\ \vdots \\ \dfrac{\partial z_1}{\partial a_n}(t) = \dfrac{-\xi^{-n}}{1+a_1\xi^{-1}+a_2\xi^{-2}+\ldots+a_n\xi^{-n}} \cdot z_1(t) \end{cases}$$

Vilket ger att $\dfrac{\partial z_1}{\partial a_{j+1}}(t) = \dfrac{\partial z_1}{\partial a_j}(t-1)$

d.v.s. $\dfrac{\partial z_1}{\partial a_{j+1}}(t+1) = \dfrac{\partial z_1}{\partial a_j}(t)$ $\quad j=1,2\ldots n$ $\quad\ldots\ldots(3.3)$

$$z(0) = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Ger begynnelsevärdena $\dfrac{\partial z_i}{\partial a_j}(0) = 0, \forall i,j$

Derivering av systemekvationerna m.a.p. b_j

$$\frac{\partial z}{\partial b_j}(t+1) = \Phi \frac{\partial z}{\partial b_j}(t) \qquad \ldots\ldots\ldots(3.4)$$

Genom att derivera $z(0) = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$ samt utnyttja 3.4 fås följande tabell.

t	$\frac{\partial z}{\partial b_1}(t)$	$\frac{\partial z}{\partial b_2}(t)$	$\frac{\partial z}{\partial b_3}(t)$	. . . . .	$\frac{\partial z}{\partial b_{n-1}}(t)$	$\frac{\partial z}{\partial b_n}(t)$
0	$\begin{matrix}1\\0\\\vdots\\0\end{matrix}$	$\begin{matrix}0\\1\\0\\\vdots\\0\end{matrix}$	$\begin{matrix}0\\0\\1\\0\\\vdots\\0\end{matrix}$		$\begin{matrix}0\\\vdots\\0\\1\\0\end{matrix}$	$\begin{matrix}0\\\vdots\\0\\1\end{matrix}$
1	$\begin{matrix}-a_1\\-a_2\\\vdots\\-a_n\end{matrix}$	$\begin{matrix}1\\0\\\vdots\\0\end{matrix}$	$\begin{matrix}0\\1\\0\\\vdots\\0\end{matrix}$		$\begin{matrix}0\\\vdots\\0\\1\\0\\0\end{matrix}$	$\begin{matrix}0\\\vdots\\0\\1\\0\end{matrix}$
2	$\Phi\begin{bmatrix}-a_1\\-a_2\\\vdots\\-a_n\end{bmatrix}$	$\begin{matrix}-a_1\\-a_2\\\vdots\\-a_n\end{matrix}$	$\begin{matrix}1\\0\\\vdots\\0\end{matrix}$		$\begin{matrix}0\\\vdots\\0\\1\\0\\0\\0\end{matrix}$	$\begin{matrix}0\\\vdots\\1\\0\\\vdots\\0\\1\\0\\0\end{matrix}$

(3:3)

Härav inses att $\frac{\partial z}{\partial b_{j+1}}(t+1) = \frac{\partial z}{\partial b_j}(t)$

och speciellt

$$\frac{\partial z_1}{\partial b_{j+1}}(t+1) = \frac{\partial z_1}{\partial b_j}(t) \quad j=1,2...n \quad ...(3.5)$$

Vid tidpunkten t+1 behöver vi alltså endast skifta ner derivatorna enligt 3.3 och 3.5 samt beräkna två nya värden

på $\frac{\partial z_1}{\partial a_1}(t+1)$ och $\frac{\partial z_1}{\partial b_1}(t+1)$ enligt 3.2 och 3.4

Beteckningar i <u>procedure</u> M

$ZA1[i] = \frac{\partial z_i}{\partial a_1}$ och $ZA[j] = \frac{\partial z_1}{\partial a_j}$

analogt för $ZB1[i]$ och $ZB[j]$

$VA[j] = \frac{\partial V}{\partial a_j}$ och $VB[j] = \frac{\partial V}{\partial b_j}$

3.b. <u>Beräkning av andraderivatmatrisen</u>

Om vi vid proceduranropet av M, sätter den värdeanropade parametern k=1, beräknas inte andraderivatmatrisen. Detta användes vid Fletcher-Powells metod, där vi inte behöver andraderivatmatrisen. I Newton-Raphsons metod måste andraderivatmatrisen beräknas. Vi har nu att:

Derivering av 3.1 ger

$$\frac{\partial^2 V}{\partial a_i \partial a_j} = 2\sum_{t=0}^{N-1} \frac{\partial z_1(t)}{\partial a_i} \cdot \frac{\partial z_1(t)}{\partial a_j} + 2\sum_{t=0}^{N-1} (z_1(t) - y_{mätt}(t)) \frac{\partial^2 z_1(t)}{\partial a_i \partial a_j}$$

(I) (II)

Om $|x>$ långt från min.punkten så är troligen gradienterna stora

Om $|x>$ nära min.punkten så är $(z_1(t) - y_{mätt}(t))$ liten

Vi försummar (II) och får då med de beteckningar, som användes i <u>procedure</u> M.

$$\frac{\partial^2 V}{\partial a_i \partial a_j} = W_{ij} = 2\sum_{t=0}^{N-1} \frac{\partial z_1(t)}{\partial a_i} \cdot \frac{\partial z_1(t)}{\partial a_j}$$

Vi får således en 2n x 2n matris. Beteckna värdet av

W_{ij} vid tidpunkten t som $W_{ij}(t) = 2\sum_{m=0}^{t} \frac{\partial z_1}{\partial a_i}(m) \cdot \frac{\partial z_1}{\partial a_j}(m)$;

$$W_{i+1,j+1}(t+1) = 2\sum_{m=0}^{t+1} \frac{\partial z_1}{\partial a_{i+1}}(m) \cdot \frac{\partial z_1}{\partial a_{j+1}}(m) = 2\sum_{m=0}^{t+1} \frac{\partial z_1}{\partial a_i}(m-1) \cdot \frac{\partial z_1}{\partial a_j}(m-1)$$

enligt 3.3 . Motsvarande för $\frac{\partial z_1}{\partial b_i}$ enligt 3.5.

Nu är $\frac{\partial z_1}{\partial a_i}(-1) = 0$ varför

$$W_{i+1,j+1}(t+1) = 2\sum_{m=0}^{t} \frac{\partial z_1}{\partial a_i}(m) \frac{\partial z_1}{\partial a_j}(m) = W_{ij}(t)$$

och då även $\frac{\partial z_1}{\partial b_i}(-1) = 0$ fås generellt att

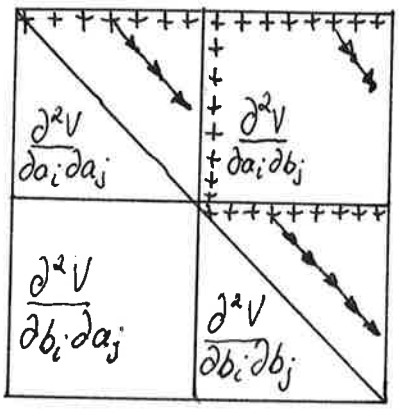
$$W_{i+1,j+1}(t+1) = W_{ij}(t) \qquad i,j=1,2....2n \qquad .....(3.6)$$

W_{ij} är symetrisk varför endast elementen i och över diagonalen behöver beräknas. Vidare behöver endast n-1 stycken skift utföras enligt 3.6, så att fram till och med tidpunkten N-n+1 beräknas endast elementen:

$W_{i,j}$	j=1,2.....2n
$W_{n+1,n+j}$	j=1,2.....n
$W_{i,n+1}$	i=2,3.....n

Således beräknas endast de i figuren markerade elementen (+) för varje tidpunkt t, där t=0,1....N-1.

(I programmet stegas t från 1 till N).

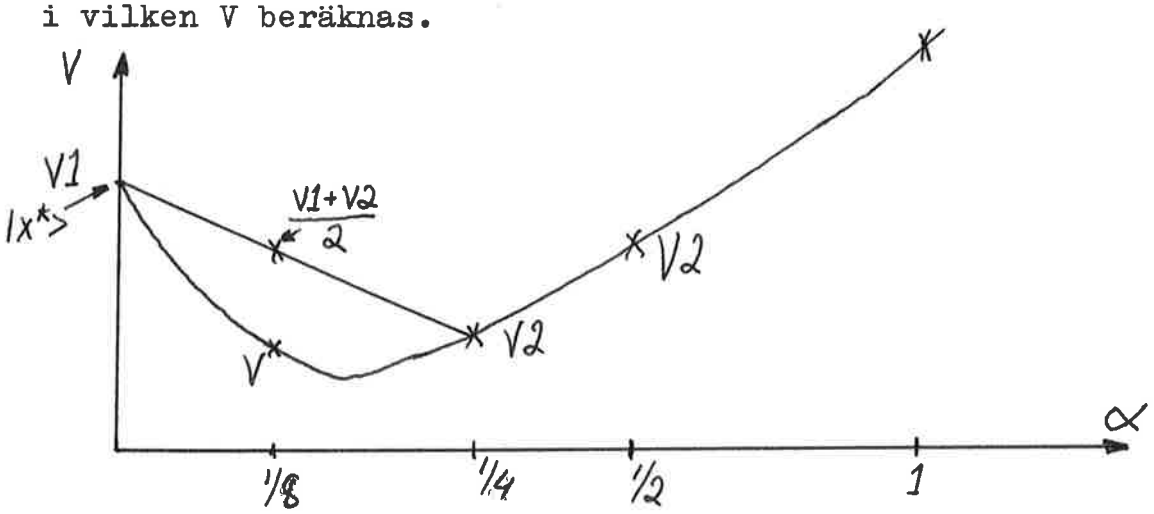


För de n-1 sista tidpunkterna utföres skift enligt pilarna i figuren.

3.c Linjär minimering

I procedure M är även inlagt beräkning av förlustfunktionen V och i samband därmed en linjär minimering längs linjen $|X> = |X^k> + \alpha\,|\delta^k>$.
Det finns ett flertal olika sätt att utföra denna minimering. Vi har valt att aproximera $V = V(\alpha)$ med en parabel, som vi bildar med hjälp av tre beräknade punkter.

Vi beskriver nu beräkningsgången hos algolproceduren procedure M. Vid start i $<x^0| = (a^0, b^0)$ sker först ingen linjär minimering, utan den "föregående" förlustfunktionen V1 sättes till 10^{36}, vilket medför att minimeringsdelen överhoppas och att gradienten och andraderivatmatrisen beräknas. Allmänt antar vi nu att vi nått punkten $<x^k| = (a^k, b^k)$ i vilken gradienten och W beräknats. Då bestäms, utanför M, en ny riktning och vi går med $\alpha = 1$ till en ny punkt i vilken V beräknas.



Om detta $V > V(|x^k>) = V1$, se figuren, sker hopp till läge PP i huvudprogrammet och vi startar från $|x^k>$ med halva steglängden ($\alpha = \alpha/2$) mot föregående gång. Halveringen fortgår tills $V \leq V1$, i vilket fall V och α lagras (V2:=V och $\alpha 2 := \alpha$). Vid den fortsatta halveringen undersöks om $V < V2$ och om så är fallet sätts V2=V och $\alpha 2 = \alpha$ (se fig.) Om $V \geq V2$ har vi nått förbi minimet utefter linjen. Här

observeras att om $V \geq \frac{V1+V2}{2}$ och vi använder V, V1 och V2 för att approximera med en parabel, når vi ett maximum och kan få ett α_{min} som är icke-positivt. I detta fall användes α 2 som steglängd. Hopp sker till läge P i huvudprogrammet, och start från $|x^k>$ med $\alpha 2$ utföres. Om $V < \frac{V1+V2}{2}$ har vi tre någorlunda vettiga punkter, nämligen V1, V2 och V. Med hjälp av dessa approximeras $V=V(\alpha)$ med en parabel och α_{min} bestämmes. Det gäller att

$$V = V_{min} + K(\alpha - \alpha_{min})^2$$
$$V1 = V_{min} + K(\alpha_1 - \alpha_{min})^2$$
$$V2 = V_{min} + K(\alpha 2 - \alpha_{min})^2$$

som ger att $$\alpha_{min} = \frac{\alpha 2^2 - \frac{V2-V1}{V-V1}\cdot\alpha^2}{2\left(\alpha 2 - \frac{V2-V1}{V-V1}\cdot\alpha\right)} \quad ;$$

Med denna steglängd startar man sedan från $|x^k>$ och når punkten $|x^{k+1}>$, i vilken V, gradienten och W beräknas, medan minimeringsdelen i procedure M hoppas över, liksom var fallet i början med punkten $|x^k>$.

En nackdel med denna minimering är att väldigt många halveringar ibland måste utföras innan $V \leq V1$.
I procedure M göres även test för att slippa spill.

Kapitel 4. Newton-Raphsons metod.

a) Kommentarer till metoden:

Den generaliserade Newton-Raphson proceduren:

$$|x^{k+1}\rangle = |x^k\rangle - G^{-1}|g^k\rangle$$

(se härledning ekvation 2.1 och Referenser)

Fördelar hos metoden:

1) Metoden invariant vid lineära transformationer av variablerna.

2) Den slutliga konvergensen är kvadratisk.

Nackdelar:

1) Ingen möjlighet att framtvinga konvergens från en dålig startpunkt.

2) Andraderivatmatrisen måste existera och måste beräknas.

3) Metoden kräver att G ej är singulär.

Den första av nackdelarna försöker vi undanröja genom att skala korrektionen med en faktor α, d.v.s. vi får: $|x^{k+1}\rangle = |x^k\rangle - \alpha^k G^{-1}|g^k\rangle$

α^k bestämmes enligt den lineära minimeringen, som beskrivs i kap. 2 och kap.3 (procedure M). Trots vår skalning och lineära minimering, kan det finnas fall, där man inte kan framtvinga konvergens (se Referenser).

Beträffande den andra nackdelen så gäller att kvadratisk konvergens inte kan uppnås utan kännedom om andraderivatorna. Kvadratisk konvergens innebär att för kvadratiska funktioner lokaliseras minimet exakt i ett ändligt antal steg, i regel n. För allmänna funktioner, som i vårt fall, gäller ofta att dessa är approximativt kvadratiska i närheten av minimet, varför metoder med kvadratisk konvergens kan vara attraktiva. Även längre

från minimet ger sådana metoder gott resultat (t.ex. för långa krökta dalar).

b) Algolprogrammet

Programmet består av 4 komponenter:

Huvudprogram

Procedure M

Procedure invers

Procedure NR

Praktiska detaljer:

Huvudprogrammet är stansat på 2 remsor, där 1/6 placeras först, innehåll deklarationer (Obs. kontrollera att fältens storlekar är tillräckliga, speciellt ym d.v.s. antalet mätvärden), och 6/6 placeras sist inom det"totala" programmet (se bifogade avskrifter). Mellan 1/6 och 6/6 insättes procedurerna M, invers och NR. Då procedure M är uppdelad på 2 remsor skall dessa placeras i ordning 2/6 resp. 3/6.

Kommentarer till procedurerna:

Procedure M beskrivs i kap. 3.

Procedure invers utför invertering av andraderivatmatrisen. (se avskriften)

Procedure NR beräknar korrektionstermerna (C[i]) till våra förlustfunktionsvariabler (a[i] och b[i]), samt utför korrigeringen av dessa. Till de så korrigerade variablerna beräknas sedan, genom anrop av procedure M, förlustfunktionen V, och om så är lämpligt även värden på gradienten (VA,VB) och andraderivatmatrisen (W). Om vi inte utfört vår lineära minimering av α inom procedure M beräknas ej ovanstående

värden, utan vi hoppar ur procedure NR till läge PP i huvudprogrammet (hoppet beodras inom procedure M). Då vi ej hoppar till PP beräknas, efter ett hopp till läge P och genomgång av procedure NR ytterligare en gång, testkvantiteten: $D=\sqrt{\sum_{i=1}^{2n}(C[i])^2}=\alpha\cdot\sqrt{\sum_{j=1}^{2n}s_j^2}=\alpha\cdot S.$

Vi skisserar nu arbetsgången i huvudprogrammet: (se bifogade avskrifter).

1) Värden på en del konstanter och startvärden läses in. Vi kommer att undersöka system med gradtal mellan r och 1. N är antalet mätvärden, delta testkvantitetens högsta värde för avslutande av iterationen, vidare sättes alfa samt s lika med 1. Nu sättes n lika med r och:

2) Procedure M anropas och våra startvärden skrivs ut. (Om man ej får utskrift "STARTVÄRDEN NEWTON/RAPHSONS METOD" har startvärdena på a och b varit alltför grovt tilltagna och vi har hoppat till läge PP.)

3) Vi lagrar våra värden på a och b, sätter V1=V, samt beräknar andraderivatmatrisens invers.

4) Därefter anropas procedure NR. Proceduren genomlöpes. Då sker:

5) Om den lineära minimeringen ej slutförts eller om V är större än V1, hoppar vi till läge PP, som används för att återhämta de i läge L (punkt 3) lagrade värdena på a och b, alfa halveras och därefter hopp till läge P (punkt 4). Start med halverat alfa från utgångspunkten blir resultatet.

6) Om ej 5 inträffar (d.v.s. α_{min} har erhållits), beräknas D och vi skriver ut de erhållna värdena. Vi testar D/alfa och D, om detta ej uppfylls sättes alfa och

s lika med 1 och vi hoppar till läge L (punkt 3). Vid uppfyllt test går vi till läge singular. Där skrivs slutliga inversen av andraderivatmatrisen ut och vi beräknar (för aktuella a och b) det erhållna systemets utsignaler för tidpunkterna 2 t.o.m. N. Dessa skrivs ut.
7) n antar nu värdet n+1 och vi återgår till punkt 2.
Om n+1 större än 1 avslutas programmet.

Kapitel 5. Fletcher-Powells metod.

För minimering med Newton-Raphsons metod gäller enligt 2.2 iterationsformeln:

$$|x^{k+1}\rangle = |x^k\rangle - \alpha^k (G^{-1})^k |g^k\rangle$$

I metoden, som är angiven av Fletcher-Powell ersätter man G^{-1} med en positivt definit symetrisk matris H. Denna matris ändras då man nått $|x^{k+1}\rangle$ med hjälp av den information man fått genom attgå i riktningen:

$$|s^k\rangle = -H^k |g^k\rangle \; ;$$

Man kan starta med vilken positivt definit matris som helst, men det kan vara lämpligt att starta med $H^0 = I$. Då sker första steget i riktningen "steepest descent". Vi har alltså att:

$$|x^{k+1}\rangle = |x^k\rangle - \alpha^k H^k |g^k\rangle$$

där den lineära minimeringen längs $|x\rangle = |x^k\rangle - \alpha \cdot H^k |g^k\rangle$ utföres enligt beskrivningen i kapitel 3 Procedur M.

Beräkning av H^{k+1}

Sätt: $|y^k\rangle = |g^{k+1}\rangle - |g^k\rangle$

$$|\sigma^k\rangle = \alpha^k H^k |g^k\rangle = \alpha^k |s^k\rangle \; ;$$

$$A^k = \frac{|\sigma^k\rangle\langle\sigma^k|}{\langle\sigma^k|y^k\rangle} \; ;$$

$$B^k = \frac{-H^k|y^k\rangle\langle y^k|H^k}{\langle y^k|H^k|y^k\rangle} \; ;$$

$$H^{k+1} = H^k + A^k + B^k \; ;$$

(Bevis: se referenser)

Då det vid körning visat sig att metoden inte alltid varit stabil, beroende på att H^{k+1} plötsligt inte blivit positivt definit, tas här med villkoret för stabilitet.

Det visas induktivt att om H^k är positivt definit

så är H^{k+1} också det om $\langle \sigma^k | y^k \rangle > 0$ (bevis:se referenser).

$$\langle \sigma^k | y^k \rangle = \langle \sigma^k | g^{k+1} \rangle - \langle \sigma^k | g^k \rangle =$$

$$= \langle \sigma^k | g^{k+1} \rangle + \underbrace{\alpha^k \langle g^k | H^k | g^k \rangle}_{\text{positivt definit}} ;$$

Om nu inte minimet hittas exakt, d.v.s. $\langle \sigma^k | g^{k+1} \rangle \neq 0$, så kan det hända att $\langle \sigma^k | g^{k+1} \rangle$ blir så mycket negativt att $\langle \sigma^k | y^k \rangle \leq 0$

Och då blir inte H^{k+1} positivt definit. Se vidare kapitel 6, svårigheter vid körningar.

Fördelar hos Fletcher-Powells metod.

1) Använder information från tidigare iterationer vid uppbyggnad av H-matrisen.

2) Inga andraderivator behöver beräknas.

3) Ingen invertering.

4) Kvadratisk konvergens.

5) Den ger "ytans" utseende vid minimipunkten, uppskattning av variansen kan göras. (H-matrisen sammanfaller efter ett antal iterationer med andraderivatmatrisens invers (G^{-1})).

Nackdelar.

Metoden blir inte stabil om den lineära minimeringen inte görs effektivt. Se kapitel 6, svårigheter vid körning.

Program

Programmet är uppbyggt av två procedurer och ett administrerande program.

Procedure M beräknar gradient, förlustfunktion och utför den lineära minimeringen med hjälp av hopp till läge P och PP. Se kapitel 3.

Procedure FP beräknar $| x^{k+1} \rangle$nytt (a,b), med hjälp av

aktuella $|x^k\rangle$, α^k, H^k och $|g^k\rangle$ enligt:

$$|x^{k+1}\rangle = |x^k\rangle - \alpha^k H^k |g^k\rangle ;$$

När $|x^{k+1}\rangle$ beräknas anropas procedure M, i vilken $V(|x^{k+1}\rangle)$ testas, och den lineära minimeringen utförs. När denna är slutförd gås hela M igenom och i slutet av procedure FP beräknas H^{k+1} enligt ovan. g^{k+1} lagras för att användas i nästa steg vid beräkning av $y^{k+1} = |g^{k+2}\rangle - |g^{k+1}\rangle$

Test av $SK1 = \langle \sigma^k | y^k \rangle$ förekommer före beräkning av H^{k+1}. Om $SK1 < 0$ blir enligt stabilitetsvillkoret inte H^{k+1} inte positivt definit. Då startar vi om i $|x^k\rangle$ och ökar α^k tills $SK1 > 0$. Det har visat sig fungera i de allra flesta fall. Mer om detta i kapitel 6. I slutet av FP beräknas dessutom $D = \sqrt{\sum_{i=1}^{n} \sigma_i^{\alpha^2}} = \alpha \sqrt{\sum_{i=1}^{n} (s_i^2)} = \alpha \cdot S$

D och S användes sedan i ett test i huvudprogrammet för att avsluta iterationen.

Beskrivning av hela programmets funktionssätt.

1) Startvärden $\langle x^0 | = (a^0, b^0)$ väljes (se kapitel 6), data läses in. H^0 sättes till enhetsmatrisen. Procedure M anropas så att förlustfunktionen V och grad beräknas i $|x^0\rangle$ (k:=1)

2) I läge L lagras $V1 = V(|x^k\rangle)$ och $|x^k\rangle$.

3) I läge P anropas procedure FP och $|x^{k+1}\rangle$ beräknas med hjälp av optimalt α. Dessutom beräknas H^{k+1} för att användas till nästa riktning $|s^{k+1}\rangle$. Intressanta variabler skrivs ut.

4) S och D testas. De är ett mått på "avståndet" till minimet. Iterationen avslutas då $S < \delta$ och $D < \delta$ (Vi har kört med $\delta = 10^{-3}$). Om så inte är fallet sättes α till 1 och vi hoppar till läge L (2) och iterationen fort-

sätter med start i $|x^{k+1}\rangle$. Läge PP är fristående och används för att återhämta de i läge L lagrade $|x^{k}\rangle$ samt halvera α och återgå till läge P. Start med halverat från punkten $|x^{k}\rangle$ blir resultatet.

Kapitel 6. Resultat och jämförlser, svårigheter vid körning. A) Resultat.

De system vi tittat på har kopplats upp på analogimaskin och impulssvaret avlästs på millimeterpapper.

Val av startpunkt $\langle x^o \rangle = (a^o, b^o)$:

Bestäm approximativt (a^o, b^o) med hänsyn till de första punkterna, $y_{mätt}(t)$, med hjälp av systemekvationerna. Korrigera (a^o, b^o) med passningsräkningar över "intressanta" delar av utsignalen. Man kan, vid bestämning av startvärden för ordning n+1, utgå ifrån minimipunktens värden för ordning n. Vi sätter då a-variablerna a_1^o t.o.m. a_n^o och b-variablerna b_1^o t.o.m. b_n^o till motsvarande värden i minimipunkten för ordning n och därefter t.ex. $a_{n+1}^o = b_{n+1}^o = 0$. Alla startvärdena på a eller b får dock inte vara noll.

För de två system vi presenterar nedan är det relativt lätt att hitta hyggliga startpunkter.

Impulssvar 1.

Vi tittar på systemet, som har överföringsfunktionen:

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

vilket ger impulssvaret $y(t) = \frac{400}{\sqrt{3}} \cdot e^{-t/2} \cdot \sin\left(\frac{\sqrt{3}}{2} t\right)$

t	0	1	2	3	4	5	6	7
$y(t)$	0	106.8	83.8	26.7	-9.85	-17.6	-10.25	-1.54
$y_{mätt}(t)$	0	107	83	27	-10	-18	-10	-2
fel i ymätt	0	+0.2	-0.8	+0.3	-0.05	-0.4	+0.25	-0.46
% fel i ymätt	0	+0.2	-0.9	+1.1	-1.5	-2	+2.5	-30

Det procentuella felet i $y_{mätt}$ växer alltså med tiden t. Nominellt ligger felet 0 och 0.8, varför förlustfunktionen i minimet kan förväntas bli omkring 1. Σ(fel)$^2 \approx$ 0.995 om inte $y_{mätt}(7)=-2$ tas med, vilket var fallet i nedanstående körningar.

Minimeringen utfördes för n=1,2,3. Se tabeller på sidorna 6:3, 6:4, 6:5, och 6:6.

Vi har alltså fått med båda metoderna

n=1 $V_{min}=1,2.10^4$

n=2 $V_{min}=1,1$

n=3 $V_{min}=0,3$

Härav sluter vi att systemet är av ordningen 2, d.v.s. vårt samplade system blir:

$$\begin{cases} z(t+1) = \begin{bmatrix} 0.785 & 1 \\ -0.367 & 0 \end{bmatrix} z(t) = e^{A} z(t) = \Phi z(t) \\ z(0) = \begin{bmatrix} 0.085 \\ 106.6 \end{bmatrix} \\ y(t) = z_1(t) \end{cases}$$

Med SUBROUTINE MATLOG, som finns på institutionen, beräknas log Φ . Vårt kontinuerliga system blir då:

$$\begin{cases} \frac{dz}{dt} = Az = \begin{bmatrix} 0.235371 & 1.876604 \\ -0.688714 & 1.237764 \end{bmatrix} z \; ; \\ z(0) = \begin{bmatrix} 0.085 \\ 106.6 \end{bmatrix} \\ y(t) = z_1(t) \end{cases}$$

n=1 Fletcher - Powell

	$V \times 10^{-4}$	a	b
Start:	1.5	-0.5	70.0000
	1.25	-0.73	69.9996
	1.219	-0.74	57.5
	1.214 809	-0.767	55.97
	1.214 807 798	-0.766 77	56.066
	1.214 807 797	-0766 765	56.0658
	1.214 807 795	-0.766 763	56.0656

Minimeringstid : 45 sek

n=1 Newton - Raphson

$V \times 10^{-4}$	a	b
1.5	-0.5	70
1.22	-0.8	50.9
1.21488	-0.7667	55.5
1.21481	-0.7669	56.05
1.21481	-0.766 76	56.065
1.21481	-0.766 76	56.066

Minimeringstid : 37 sek

V	a_1	a_2	b_1	b_2	
168	-0.7	0.3	0	107	Fletcher-Powell
95	-0.74	0.304	0.000 04	107.000 2	
1.97	-0.7876	0.3677	0.000 02	107.000 6	
1.3	-0.7868	0.3683	-02	106.61	
1.068	-0.7845	0.3671	0.07	106.63	
1.067 464 6	-0.784 633	0.367 332	0.084	106.609	
1.067 464 2	-0.784 634	0.367 332	0.085	106.608	
168	-0.7	0.3	0	107	Newton-Raphson
1.61	-0.788	0.3670	0.1	106.4	
1.067 47	-0.784 62	0.367 31	0.08 47	106.609	
1.067 46	-0.784 63	0.367 33	0.085 3	106.608	

N = 2

Fletcher-Powell — 105 sek

Newton-Raphson — 54 sek

Minimeringsbild

h:9

N=3 Newton-Raphson

V	a_1	a_2	a_3	b_1	b_2	b_3
1626	-0.3	0.1	0.3	0	107	30
44	-0.1	-0.1	0.21	-0.2	106.91	71.6
0.46	-0.07	-0.2	0.27	-0.009	107	75.5
0.271935	-0.3092	-0.0109	0.1798	-0.01362	107.0061	49.96
0.271934	-0.30942	-0.01072	0.17970	-0.01359	107.00583	49.938
0.271934	-0.30941	-0.01073	0.17970	-0.01359	107.00583	49.93%

Minimeringstid: 850 sek 18 st. iterationer

Här har endast mellanpunkterna, sedan ungefär var 5:e.

6:5

n = 3 Fletcher - Powell 6:6

V	a_1	a_2	a_3	b_1	b_2	b_3
1626	-0.3	0.1	0.3	0	107	30
141	-0.39	-0.02	0.2	-0.0003	106.999886	30.0005
69	-0.47	0.05	018	0.00004	107.0003	30.002
16	-0.488	011	0117	0.0003	107.0006	30.003
3	-0.486	0.126	0.114	-1.05	106.48	31.06
034	-0.489	0.125	0.117	-0.015	107.006	31.5
0.32	-0.42	0.07	0.14	-0.001	107.07	38
0.30	-0.36	0.08	0.16	-0.019	107.065	44
0.278	-0.33	0.002	0.173	-0.017	107.05	48
0.2722	-0.310	-0.0100	0.1793	-0.009	107.001	49.86
0.2719	-0.309408	-0.010735	0.179705	-0.013589	107.005868	49.939770

Minimeringstid : 640 sek. 33 st. iterationer

Här har endast medtagits de 5 första punkterna, sedan ungefär var 5:e.

$$\det[\lambda I - A] = \lambda^2 + 1.002\lambda + 0.999 = 0 \qquad (6:7)$$

Genom att göra transformationen x=Tz kan vi få A på normalform enligt:

$$T \cdot \begin{bmatrix} 0.235 & 1.877 \\ -0.689 & -1.238 \end{bmatrix} T^{-1} = \begin{bmatrix} -1.002 & 1 \\ -0.999 & 0 \end{bmatrix}$$

Dessutom villkoret att

$$y(t) = [1 \;\; 0]\, T^{-1} x(t) = x_1(t)$$

Enkla räkningar ger att

$$T = \begin{bmatrix} 1 & 0 \\ 1.237 & 1.877 \end{bmatrix};$$

d.v.s. vårt kontinuerliga system blir:

$$\begin{cases} \dfrac{dx}{dt} = TAT^{-1}x = \begin{bmatrix} -1.002 & 1 \\ -0.999 & 0 \end{bmatrix} X \\ X(0) = TB = \begin{bmatrix} 1 & 0 \\ 1.237 & 1.877 \end{bmatrix} \begin{bmatrix} 0.085 \\ 106.6 \end{bmatrix} = \begin{bmatrix} 0.085 \\ 200.2 \end{bmatrix} \\ y(t) = x_1(t) \end{cases}$$

Vårt $G(s) = 200\,\dfrac{0.0005 \cdot s + 1.001}{s^2 + 1.002 \cdot s + 0.999}$;

Det verkliga systemet är $G(s) = 200\,\dfrac{1}{s^2 + s + 1}$;

d.v.s. max. fel $\sim$ 0,2% i systemparametrarna.

Impulssvar 2.

Vi tittar på systemet som har överföringsfunktionen

$$G(s) = 100\,\frac{s-1}{s^2+s+1};$$

vilket ger impulssvaret

$$y(t) = 200\, e^{-t/2} \cos\left(\frac{\sqrt{3}}{2}t + \frac{\pi}{3}\right);$$

t	0	1	2	3	4	5	6
$y(t)$	100	-40.8	-68.8	-39.1	-5.4	10.1	9.99
$y_{mätt}(t)$	100	-45	-67.5	-34	-2	11	9
fel i ymätt	0	-4.2	+1.3	+5.1	+3.4	-0.9	-0.99
% fel i ymätt	0	-10	+2	+13	+63	+9	-11

Vi har alltså här ett större mätfel än vid impulssvar 1. $\Sigma(\text{fel})^2 \sim 50$. (Troligen har fel på skrivaren inträffat vid körning på analogimaskinen).

Se tabeller på sidorna 6:9, 6:10 och 6:11, dessa visar resultatet från körningarna.

Vi har alltså fått med båda metoderna

n=1 $V_{min} = 7.10^3$

n=2 $V_{min} = 0{,}7$

n=3 $V_{min} = 0{,}1$

Härav sluter vi att systemet är av ordning 2, d.v.s. vårt samplade system blir:

$$\begin{cases} z(t+1) = \begin{bmatrix} +0.736 & 1 \\ -0.348 & 0 \end{bmatrix} z(t) = e^{A} z(t) = \Phi z(t) \\ z(0) = \begin{bmatrix} 99.9 \\ -118.3 \end{bmatrix} = B \\ y(t) = z_1(t) \end{cases}$$

A= log Φ beräknas med SUBROUTINE MATLOG och

$$\begin{cases} \dfrac{dz}{dt} = \begin{bmatrix} 0.188316 & 1.945903 \\ -0.677179 & -1.293869 \end{bmatrix} z \\ z(0) = \begin{bmatrix} 99.9 \\ -118.3 \end{bmatrix} \\ y(t) = z_1(t) \end{cases}$$

n=1 Fletcher - Powell

$V \cdot 10^{-3}$	a	b
9	0.5	100.0000
7.05	0.22	99.9992
7.042	0.20	102.9
7.041 189 4	0.19881	102.34
7.041 189 312 9	0.198847	102.3272
7.041 189 312 9	0.198844	102.3279

Minimeringstid: 39 sek

n=1 Newton - Raphson

$V \cdot 10^{-3}$	a	b
9	0.5	100
7.044	0.2	102.57
7.0412	0.1986	102.54
7.041 19	0.1990	102.3321
7.041 19	0.1988	102.3318

Minimeringstid: 29 sek

V	a_1	a_2	b_1	b_2	
598	-0.7	0.30	100.0000	-102.0000	Fletcher-Powell
292	-0.64	0.37	99.9996	-100.0003	
281	-0.66	0.39	99.9994	-100.002	
3.6	-0.76	0.36	94.5	-115	
1.2	-0.737	0.349	100.1	-118.0	
0.676	-0.7357	0.34771	99.95	-118.37	
0.675 33	-0.735 59	0.347 72	99.93	-118.341 1	
0.675 303 03	-0.735 614	0.347 702	99.928 6	-118.3418	
0.675 302 97	-0.735 615	0.347 704	99.928 4	-118.3417	
598	-0.7	0.3	100	-100	Newton-Raphson
2.3	-0.733	0.35	99.920	-117.7	
0.675-39	-0.735 68	0.347 73	99.926	-118.345	
0.675 303	-0.735 61	0.347 70	99.928 42	-118.341 71	
0.675 303	-0.735 61	0.347 70	99.928 43	-118.341 77	

Fletcher-Powell Newton-Raphson

Minimering på 129 sek 73 sek

n = 2

6:10

$n = 3$ Newton - Raphson

V	a_1	a_2	a_3	b_1	b_2	b_3
57	−0.3	0	0.2	100	−73	−55
0.8	−0.46	0.1	0.1	99.98	−91	−33
0.5	−1.3	0.8	−0.2	99.998	−175	68
0.29	−2	1.3	−0.5	99.996	−249	155
0.17	−1.2	0.67	−0.15	100.002	−163	53
0.12	−1.57	0.96	−0.29	100.0019	−202	100
0.1146	−1.66	1.02	−0.31	100.0040	−211	109
0.11453	−1.621	0.997	−0.302	100.00435	−207.2	105.4
0.114528	−1.626	1.0009	−0.3036	100.00429	−207.65	105.9
0.114528	−1.62513	1.00020	−0.30326	100.00430	−207.54843	105.7989
0.114528	−1.62513	1.00020	−0.30396	100.00430	−207.54849	105.7941

Minimeringstid: 400 sek 10 iterationer

6:11

$$\det[\lambda I - A] = \lambda^2 + 1.056\lambda + 1.084 = 0 \qquad (6{:}12)$$

Analogt med tidigare (se sid 6:7) görs transformationen $x = Tz$.

$$\begin{cases} \dfrac{dx(t)}{dt} = TAT^{-1} = \begin{bmatrix} -1.056 & 1 \\ -1.084 & 0 \end{bmatrix} X(t) \\ X(0) = TB = \begin{bmatrix} 1 & 0 \\ 1.243 & 1.946 \end{bmatrix} \begin{bmatrix} 99.9 \\ -118.3 \end{bmatrix} = \begin{bmatrix} 99.9 \\ -107 \end{bmatrix} \\ y(t) = X_1(t) \end{cases}$$

Vårt $G(s) = 100 \cdot \dfrac{0.999 \cdot s - 1.07}{s^2 + 1.056 \cdot s + 1.084}$

Det verkliga systemet är: $G(s) = 100 \cdot \dfrac{s-1}{s^2+s+1}$;

d.v.s. max. fel 8% i systemparametrarna. Så stort är dock inte felet, ty de värden, som insatts i programmet, härrör inte från $G(s) = 100 \cdot \dfrac{s-1}{s^2+s+1}$; (se föregående).

Med båda pulssvaren har vi tydligen fått bättre noggrannhet i systemparametrarna än noggrannheten i $y_{mätt}$. Detta verkar inte helt orimligt, då vi anpassar utsignalen från systemet så att V blir minimum. Om felen i $y_{mätt}$ har medelvärdet noll, kommer vårt impulssvar att bättre svara mot det verkliga systemets än det uppmätta impulssvaret.

Impulssvar 1	max. fel	0,2%
Impulssvar 2	max. fel	8%

Uppenbarligen beror denna stora skillnad på de stora felen i $y_{mätt}$ för impulssvar 2 och att dessa har en olycklig spridning.

Vid uppkoppling av

$$G(s) = 100 \cdot \frac{0{,}999s - 1{,}07}{s^2 + 1{,}056s + 1{,}084}$$

på analogimaskinen och körning av systemet fick vi ett resultat, som visar att vår förmodan om felaktiga indata för impulssvar 2 var riktig. (se sida 6:14) Värdena för

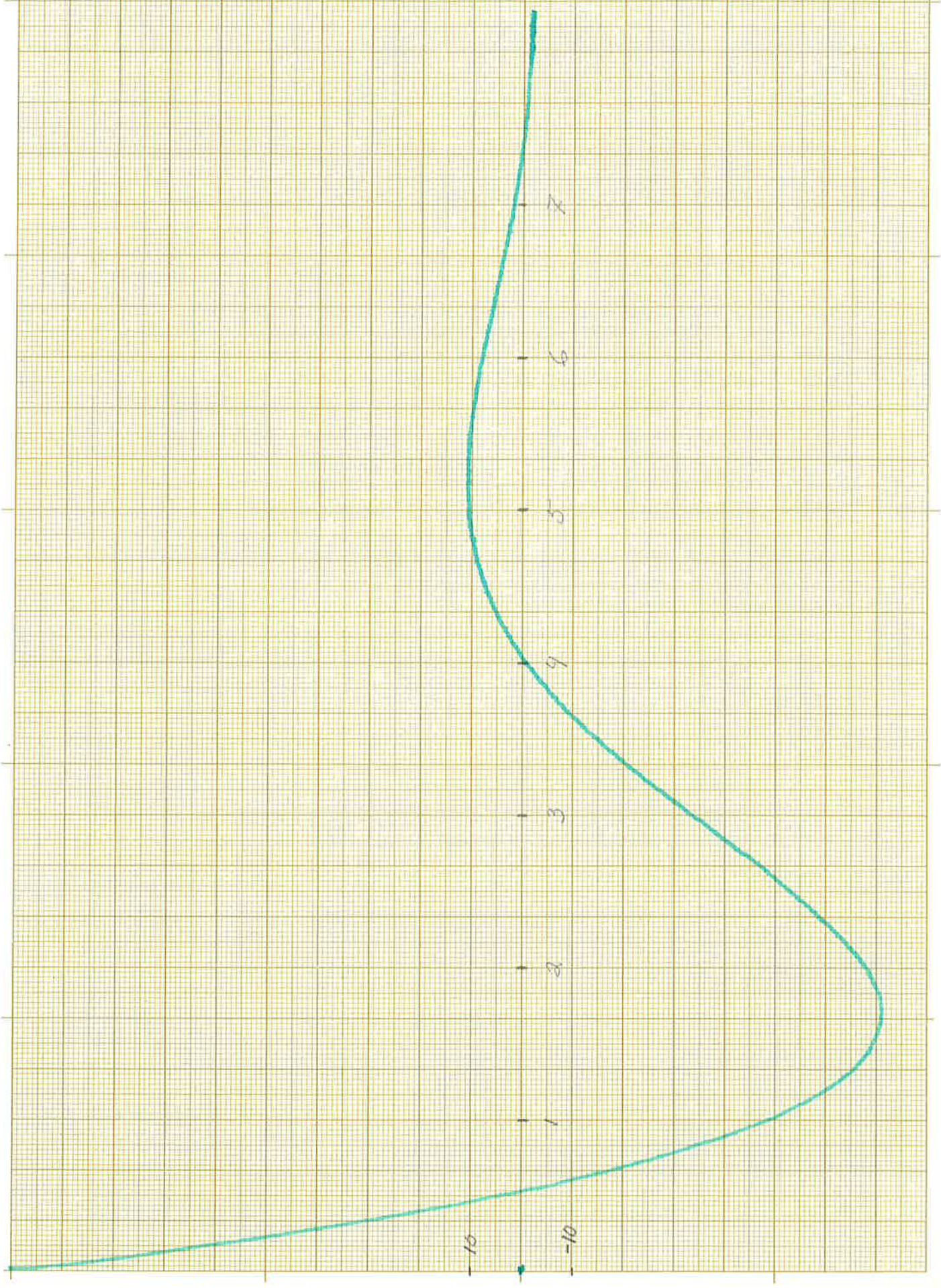
$$G(s) = 100 \cdot \frac{0{,}999s - 1{,}07}{s^2 + 1{,}056s + 1{,}084}$$

överensstämmer nämligen mycket väl till de indata vi hade för impulssvar 2 ($y_{mätt}(t)$).

För impulssvar 1 se sida 6:15.

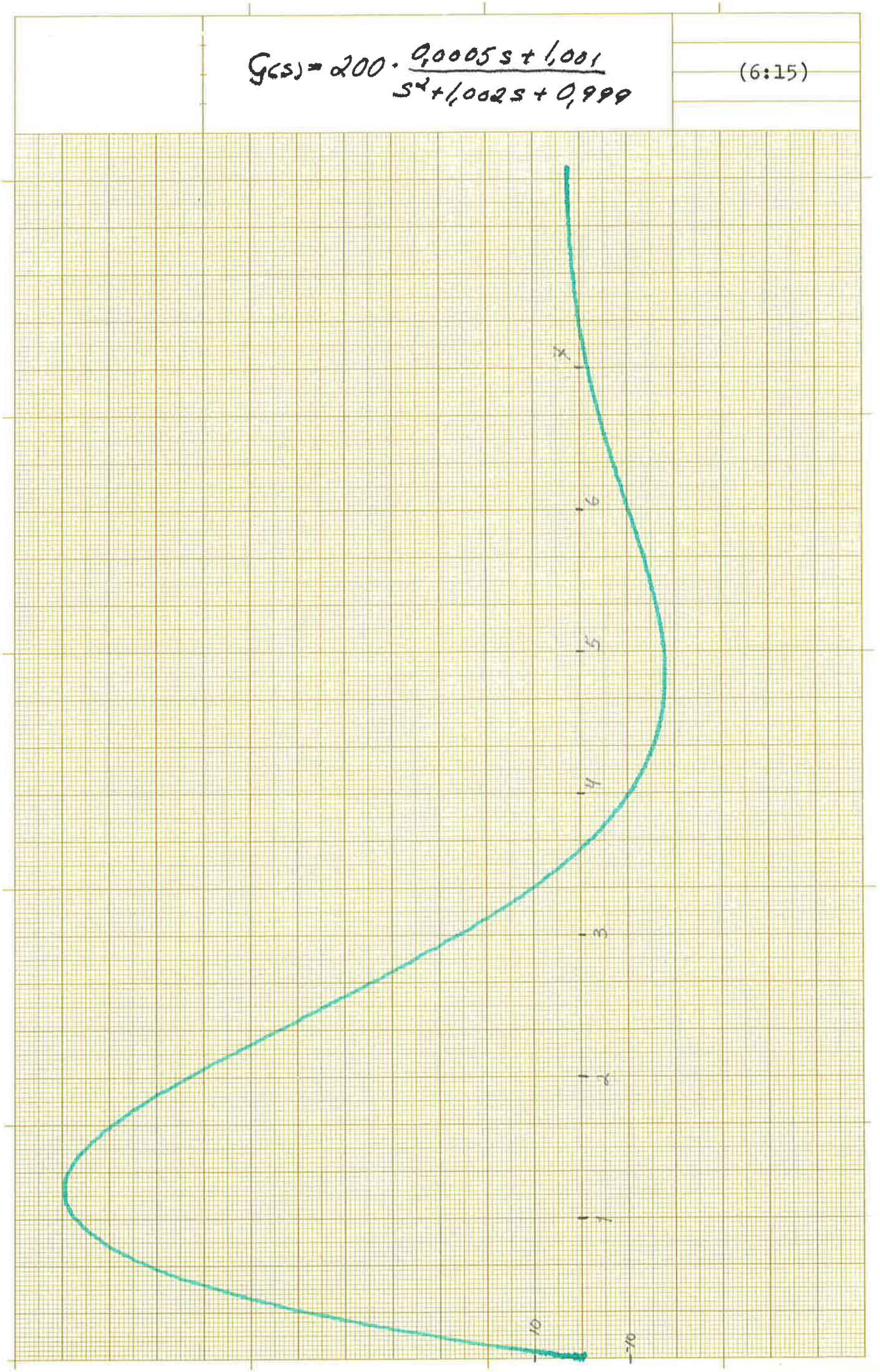
(6:14)

$$G(s) = 100\,\frac{0.9995 - 1.07}{s^2 + 1.056\,s + 1.084}$$



$$G(s) = 200 \cdot \frac{0{,}0005\,s + 1{,}001}{s^2 + 1{,}002\,s + 0{,}999}$$

(6:15)



B. Jämförelse mellan metoderna Fletcher-Powell och Newton-Raphson.

Vi ser nu på minimeringstiderna (antalet steg är satt inom parentes).

1: Start i närheten av minimipunkten

Impulssvar 1 (tid i sekunder)

n	1	2	3
F-P	45(6)	105(6)	640(33)
N-R	37(5)	54(3)	850(18)

Impulssvar 2

n	1	2	3
F-P	39(5)	122(8)	
N-R	29(4)	73(4)	400(10)

Här ses att för n=1,2 är Newton-Raphsons metod snabbare än Fletcher-Powells. För n=3 är Fletcher-Powells metod snabbare trots att iterationen sker i fler steg än med Newton-Raphsons metod.

2: Start långt ifrån minimipunkten

n=2, Start i $(a^{o}, b^{o}) = (1,1,1,1)$

Här tog minimeringen lika lång tid (ca:5 min.) för bägge metoderna. Detta jämföres med n=2 under 1 där Fletcher-Powells metod tog dubbelt så lång tid som Newton-Raphsons.

3: Högre ordningstal

Vid körning med ett större exempel (se kapitel 7), fick vi vid ordningstalet n=4 och start långt ifrån minimipunkten att för en bit i början av iterationen, så var Fletcher-Powells metod ca:5 gånger snabbare än Newton-Raphsons.

	V_{start}	$V_{jämförlse}$	Tid (min.)
Newton-Raphson	5,8	1,7	ca: 53
Fletcher-Powell	5,8	1,2	" 11

De olikheter vi har mellan de båda metoderna beror på,att vi i Newton-Raphsons metod måste beräkna andraderivatmatrisen samt dess invers, vilket blir tidsödande vid respektive många mätpunkter och högt ordningstal. I Fletcher-Powells metod bygger vi upp matrisen H med hjälp av den information vi har hos gradienterna. Vi använder också enhetsmatrisen, som startmatris. D.v.s. vid första steget går vi i "steepest descent" . Detta ger tydligen kortare körtider vid höga ordningstal, många mätpunkter eller start långt från minimipunkten, för Fletcher-Powells metod. När man har låga ordningstal eller start i närheten av min. punkten så går Newton-Raphsons metod snabbare.

C. Svårigheter vid körningar.

1) Det kan i vissa fall vara besvärligt att välja startpunkt. Om startpunkten hamnar väldigt långt från minimipunkten uppstår spill vid beräkning av förlustfunktionen. Detta har vi eliminerat genom test i procedure M.

2) Newton-Raphsons metod.

Vid denna metod krävs att andraderivatmatrisens invers (W^{-1}) skall vara positivt definit. Att matrisen ej blir positivt definit inträffar förhållandevis sällan. Vår approximativa andraderivata utgör en förbättring i detta avseende.

Om matrisen ej är positivt definit, medför det att vi går i en sådan rikting att förlustfunktionen är växande. Man hittar därför, genom halvering av α , inget α -värde, som ger en mindre förlustfunktion än utgångspunktens. Vid körning på datamaskin skulle detta medföra att α halverades tills det fick värdet noll i maskinen. För att undanröja detta har ett test på α lagts in i programmet vid läge P före anropandet av procedure NR. Vid för lågt α får vi utskriften ALFA FÖR LITET, samt hopp till läge singular.

För vissa startpunkter fås ej konvergens (se kapitel 4 och referenser), vilket elimineras genom att välja ny startpunkt.

3) Fletcher-Powells metod.

Vid minimering med denna metod kan H-matrisen helt plötsligt bli icke-positivt definit. Fortsätter iterationen då går man ju i ökande riktning (jämför ovanstående). Vi får således ingen mindre förlustfunktion, d.v.s. α hal-

veras tills maskinens α =0. Då gäller att $\sigma^k = \alpha^k \cdot s^k = 0$ och således $SK1 = \langle \sigma^k | g^{k+1} \rangle = 0$

Vid bildande av H^{k+1} fås då division med noll.

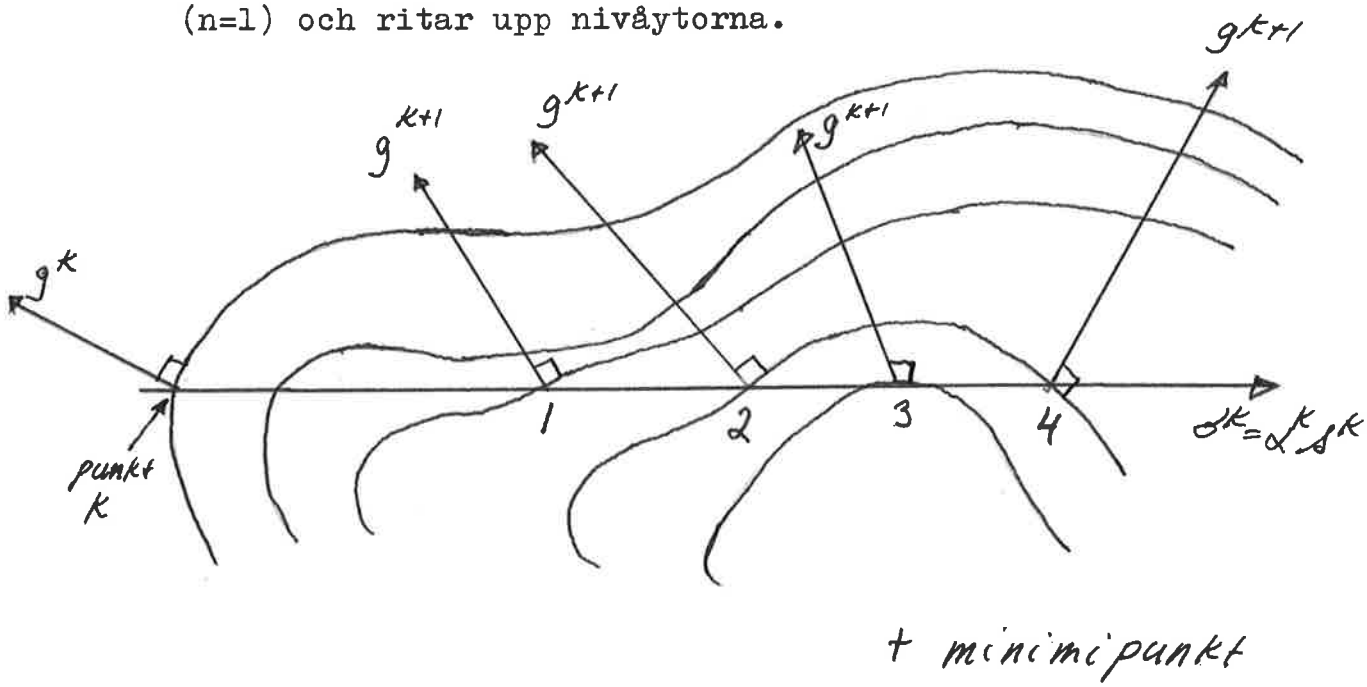
Orsaken till att H^{k+1} kan bli icke positivt definit ligger i villkoret för stabilitet, se beskrivningen av Fletcher-Powells metod i kapitel 5. Villkoret är att

$$SK1 = \langle \sigma^k | y^k \rangle = \langle \sigma^k | g^{k+1} - g^k \rangle > 0$$

$$SK1 = \langle \sigma^k | g^{k+1} \rangle - \underbrace{\langle \sigma^k | g^k \rangle}_{>0}$$

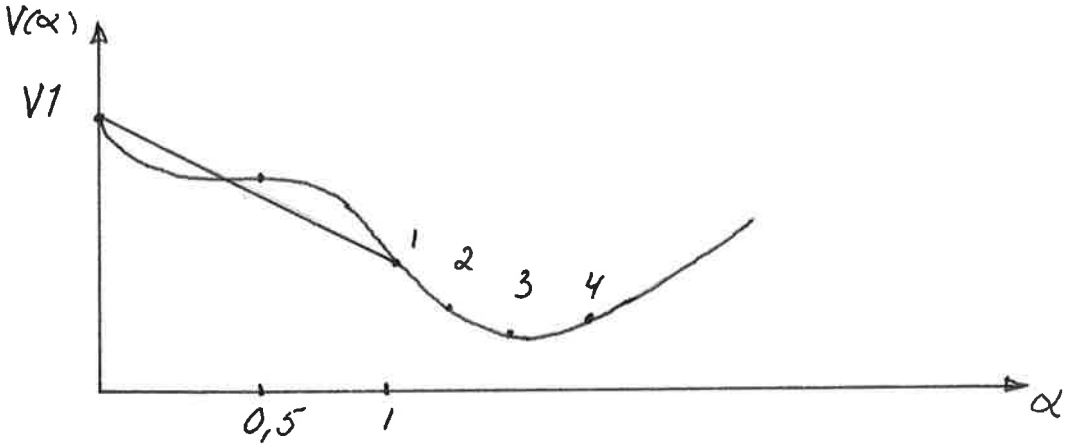
Vi vill alltså att $\langle \sigma^k | g^{k+1} \rangle \geq 0$ för att $SK1 > 0$ säkert skall gälla.

Fenomenet uppträder vid högre ordningstal $n \geq 2$, varför den här framställningen endast skall betraktas som ett försök till illustration. Det är ju omöjligt att tänka sig hur nivåytorna ser ut för t.ex. n=2 (d.v.s. en funktion av 4 variabler). Vi tänker oss en funktion av två variabler (n=1) och ritar upp nivåytorna.



+ minimipunkt

Vi har kommit till punkten k och går i riktning s^k. I denna riktning ser vi att $V=V(\alpha)$ ser ut som

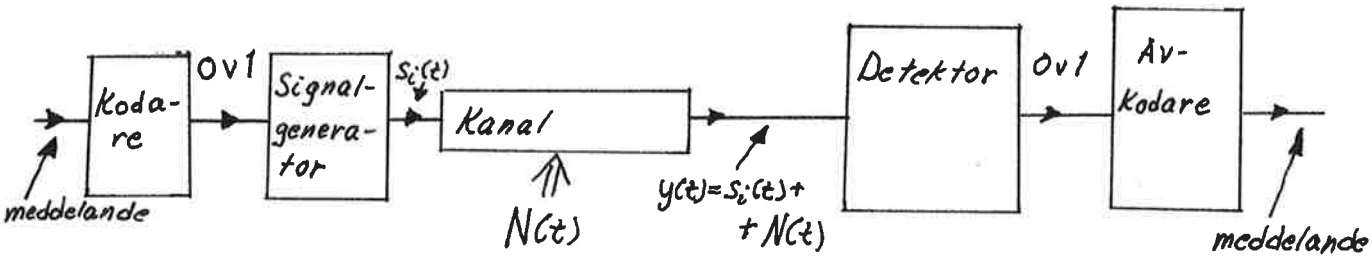


Först sättes α =1 och vi går ut till punkten 1. Enligt vår minimeringsmetodik kommer vårt optimala α att bli 1 (se procedure M). Vi skall alltså i denna punkt räkna ut en ny H-matris. Här är dock $\langle \delta^k | g^{k+1} \rangle < 0$ och risk för att $SK1 < 0$ föreligger. Om $SK1 < 0$ ökas α (se procedure FP) och vi kommer i punkter där $\langle \delta^k | g^{k+1} \rangle$ ökar. Ökningen av α fortgår tills $SK1 > 0$, och således tills H^{k+1} blir positivt definit. I verkligheten ser funktionerna mycket konstigare ut. Det har hänt att SK1 blivit mer och mer negativt för att sedan gå mot noll och slutligen bli positiv.Det hela har dock fungerat för samtliga fall utom ett. Vid körning med ett större exempel (n=4) (se kapitel 7) gick iterationerna fint tills det att V blev ungefär 0.89 (V_{start}= 5.8). Detta gjordes med 13 steg. Efter ytterligare 21 steg hade V sjunkit till 0.88. Vi hade tydligen kommit in i någon slags svagt sluttande "dalgång". När V var 0.88 blev $SK1 < 0$, och vid den fortsatta stegningen blev SK1 mer negativt för att sedan bli mindre negativt och slutligen anta konstant negativt värde för varje ökning av α . Här fungerade alltså inte den ovan beskrivna metodiken. Lämpligt är att i denna punkt starta om, och således gå i riktning "steepest descent".

Vi fortsatte från denna punkt (V=0.88) med Newton-Raphsons metod och fullföljde minimeringen (V_{min}= 0.16).

Kapitel 7. Större exempel.

På institutionen för teletransmissionsteori sysslar man bl.a. med signaldetektering. Grovt förenklat ser problemet ut enligt följande (mer detaljerat se referenser).



Meddelandet, som skall överföras kan föreligga i antingen analog eller digital form. Kodaren transformerar meddelandet till 0 eller 1, som i sin tur styr en signalgenerator. Signalgeneratorn ger $s_0(t)$ ut om den får en nolla på ingången och ger $s_1(t)$ ut om en etta på ingången. Apriorisannolikheterna är $P(s_0)$ och $P(s_1)$. Signalerna $s_0(t)$ och $s_1(t)$ med varaktigheten T är deterministiska. I kanalen distorderas signalen och den mottagna signalen blir $y(t) = s_i(t) + N(t)$, där $N(t)$ är normalfördelat vitt brus. Vid slutet av varje signalintervall, d.v.s. vid tidpunkten T, skall mottagaren, som förutsättes vara i perfekt synkronism med sändaren, på basis av den observerade signalen $y(t) = s_i(t) + N(t)$ besluta vilken av de två signalerna som har sänts.

Signalenergien

$$E_0 = \int_0^T s_0^2(t)\,dt$$

$$E_1 = \int_0^T s_1^2(t)\,dt$$

N_0 = brusets spektraltäthet.

Avsikten är att bestämma en mottagare, som minimerar totala felsannolikheten. Man visar (se referenser) att en optimal mottagare, som ger minsta felsannolikhet, skall från den mottagna signalen beräkna storheten

$$\xi = \int_0^T y(t)\,[s_1(t) - s_0(t)]\,dt$$

och jämföra denna med en konstant

$$\zeta = \frac{1}{2}(E_1 - E_0) + N_0 \ln \frac{P(s_0)}{P(s_1)}$$

Storheten ζ kallas för beslutströskeln. Är $\xi > \zeta$ tolkas detta som att signalen s_1 blivit utsänd. Är $\xi < \zeta$ har med största sannolikhet $s_0(t)$ blivit utsänd.

Signalanpassade filter.

För att beräkna korrelationsintegralen

$$\xi = \int_0^T y(t)\,[s_1(t) - s_0(t)]\,dt$$

kan man använda sig av antingen en korrelator (se referenser) eller ett signalanpassat filter. Signalen s(t) är begränsad i tiden, intervall (0,T). Man kan då beräkna korrelationsintegralen med hjälp av ett realiserbart filter. Ett filter med impulssvaret $h_0(t)$, som utgöres av ett s(t) spegelvänd i tiden $h_0(t) = a.s(T-t)$, kallas ett signalanpassat filter. Låter vi en godtycklig funktion y(t) passera ett sådant filter erhålles utfunktionen $\hat{y}(t)$

$$\hat{y}(t) = \int_{-\infty}^{\infty} y(u)\,h_0(t-u)\,du = a\int_{-\infty}^{\infty} y(u)\cdot s(T+u-t)\,du$$

Vid tidpunkten t=T, då beslutet om vilken signal som har sänts, skall fattas blir $\hat{y}(T) = a\cdot\int_{-\infty}^{\infty} y(u)\cdot s(u)\,du$

vilket ju bortsett från konstanten a är vår sökta korrelationsintegral.

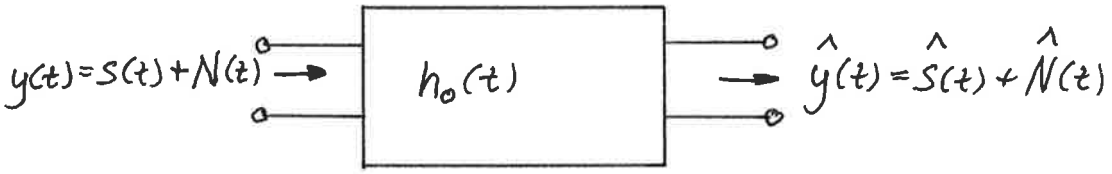
Nu är signalen $y(t) = s(t) + N(t)$; $\hat{y}(t) = \hat{s}(t) + \hat{N}(t)$;

$$\hat{s}(t) = a \int_{-\infty}^{\infty} s(u) \cdot s(T+u-t)\, du ;$$

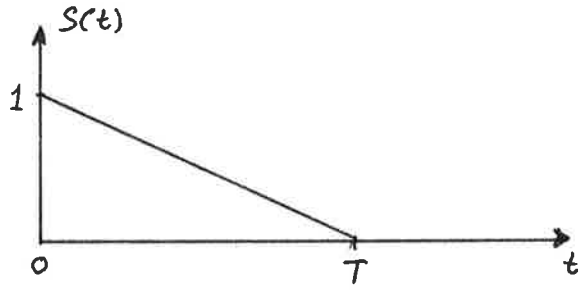
För t=T har $\hat{s}(t)$ sitt maximum

$$\hat{s}(T) = a \int_{-\infty}^{\infty} s(u) \cdot s(u)\, du = a \cdot E$$

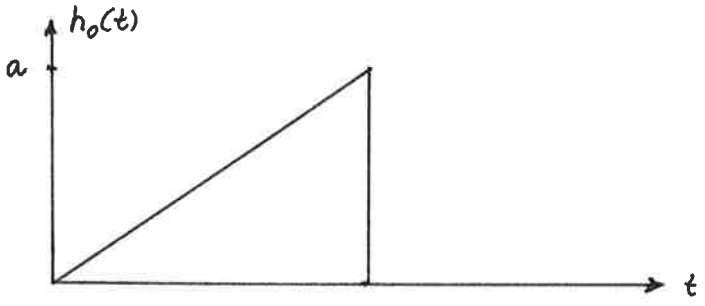
d.v.s. ett maximum som är proportionellt mot signalens energi



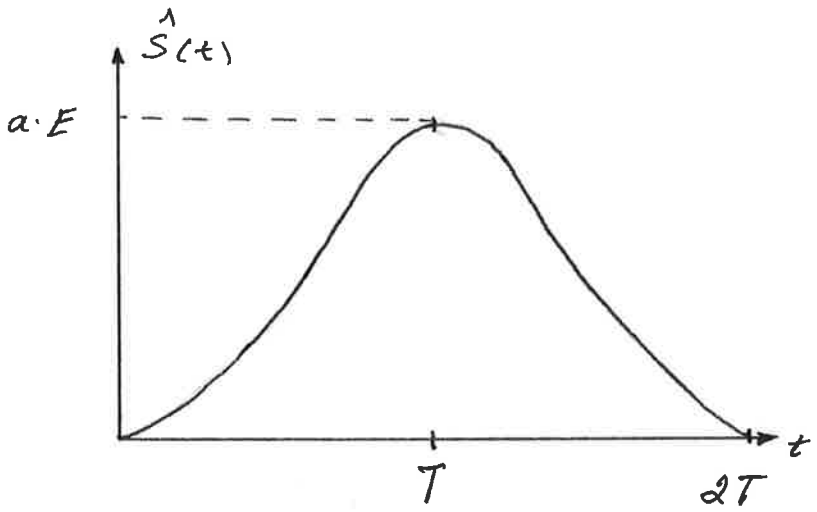
Om vi exempelvis har att s(t) ser ut som



så får vi att $h_0(t) = a.s(T-t)$ ser ut som



och vi får $\hat{s}(t)$

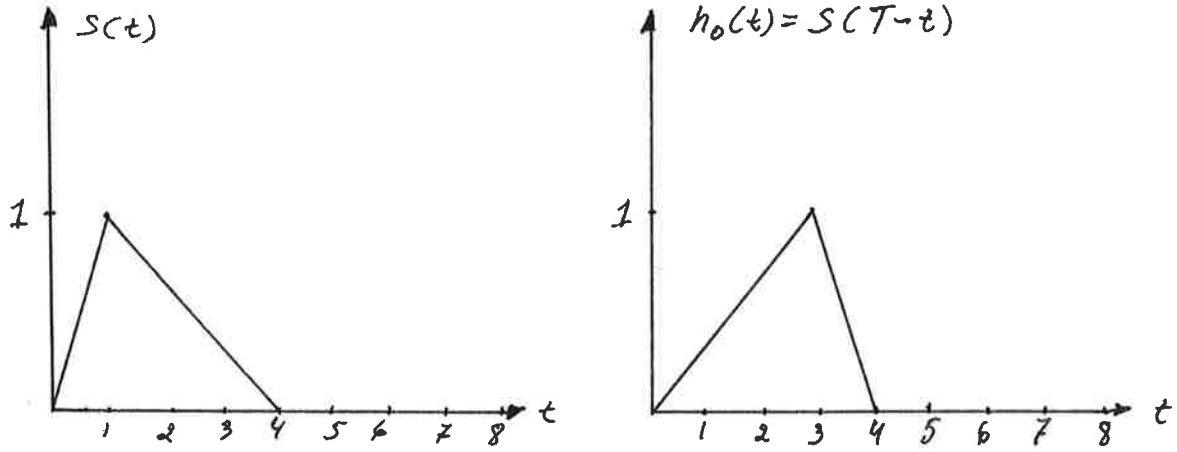


En intressant egenskap hos det signalanpassade filtret är att det maximerar signal-brusförhållandet vid filtrering av en känd signal i vitt brus. Signal-brusförhållandet = $\xi(t)$

$$\xi(t) = \frac{|\hat{s}(t)|^2}{E\{\hat{N}^2(t)\}}$$

där $E\{\hat{N}^2(t)\}$ är medeleffekten hos bruset efter filtret. Man kan visa att $\xi(T)$ antar sitt största värde om man använder ett filter med impulssvaret $h_o(t) = a.s(T-t)$. Viktigt i sammanhanget är att det inte är fråga om signalens utseende, som spelar någon roll. Det är endast signalens energi, vilket betyder att skilda signaler ger samma maximala värde på $\xi(T)$ så länge de har samma energi.

Man har valt en utsignal s(t) från sändaren och således ett spegelvänt impulssvar enligt figuren



Vår uppgift är nu att hitta ett system med impulssvaret $h_o(t) = s(T-t)$.

Kravet är att den maximala signalamplituden för $t \geq 4$ skall vara mindre än 10% av maximala amplituden för hela impulssvaret. Vi har använt oss av 33 mätpunkter. Det vore bättre att använda fler punkter, men det ställer svårt med hänsyn till tiden.

Resultat av körningar.

k = max amplitud för $t < 4$ / max. ampl. för hela impulssvaret

n	V_{min}	k %	Minimeringstid	Metod
3	0.37	19	20 min.	N-R
4	0.16	9	2 tim. (x)	F-P och N-R
5	0.068	5	1 tim. (xx)	F-P

(x) Dåligt val av startpunkt,vilket medförde att minimeringen med Fletcher-Powells metod"spårade ur". Detta inträffade efter ca. 1 tim. Se vidare kap.6,svårigheter vid körningar.

(xx) Bra val av startpunkt.Vi tog minimipunkten för n = 4 och satte $a_5 = b_5 = 0$.

Impulssvarens utseende i minimipunkterna se diagram på (7:6).

Vi stannade vid n = 5 och vårt samplade system ser ut som

$$z(t+h) = \begin{bmatrix} 3.897 & 1 & 0 & 0 & 0 \\ -6.427 & 0 & 1 & 0 & 0 \\ 5.585 & 0 & 0 & 1 & 0 \\ -2.548 & 0 & 0 & 0 & 1 \\ 0.487 & 0 & 0 & 0 & 0 \end{bmatrix} z(t) = \Phi z(t) = e^{Ah} z(t)$$

$$z(0) = \begin{bmatrix} 0.027 \\ -0.080 \\ 0.236 \\ -0.324 \\ 0.194 \end{bmatrix} = B$$

Med SUBROUTINE MATLOG beräknas

$$A = \frac{1}{h} \log \Phi$$

$$h = 0{,}25 \text{ sek.}$$

Vi har då det kontinuerliga systemet

$$\begin{cases} \frac{dz}{dt} = Az \\ z(0) = B \\ y(t) = z_1(t) \end{cases}$$

Men enligt kap.1 är detta detsamma som

$$\begin{cases} \frac{dz}{dt} = Az + Bu \\ y(t) = Cz(t) = [1{,}0{,}0{,}0] z(t) = z_1(t) \\ u(t) = \delta(t) \end{cases}$$

som dock inte är på normalform. Med PROGRAM LAPLACE (K. Eklund) erhålles direkt systemets överföringsfunktion samt poler och nollställen.

$$G(s) = \frac{0.072s^4 - 0.487s^3 + 4.527s^2 - 6.429s + 20.038}{s^5 + 2.877s^4 + 11.286s^3 + 17.696s^2 + 20.954s + 9.878}$$

Poler: -0.752788

$-0.398986 \pm i2.54925$

$-0.663172 \pm i1.23738$

Nollställen:

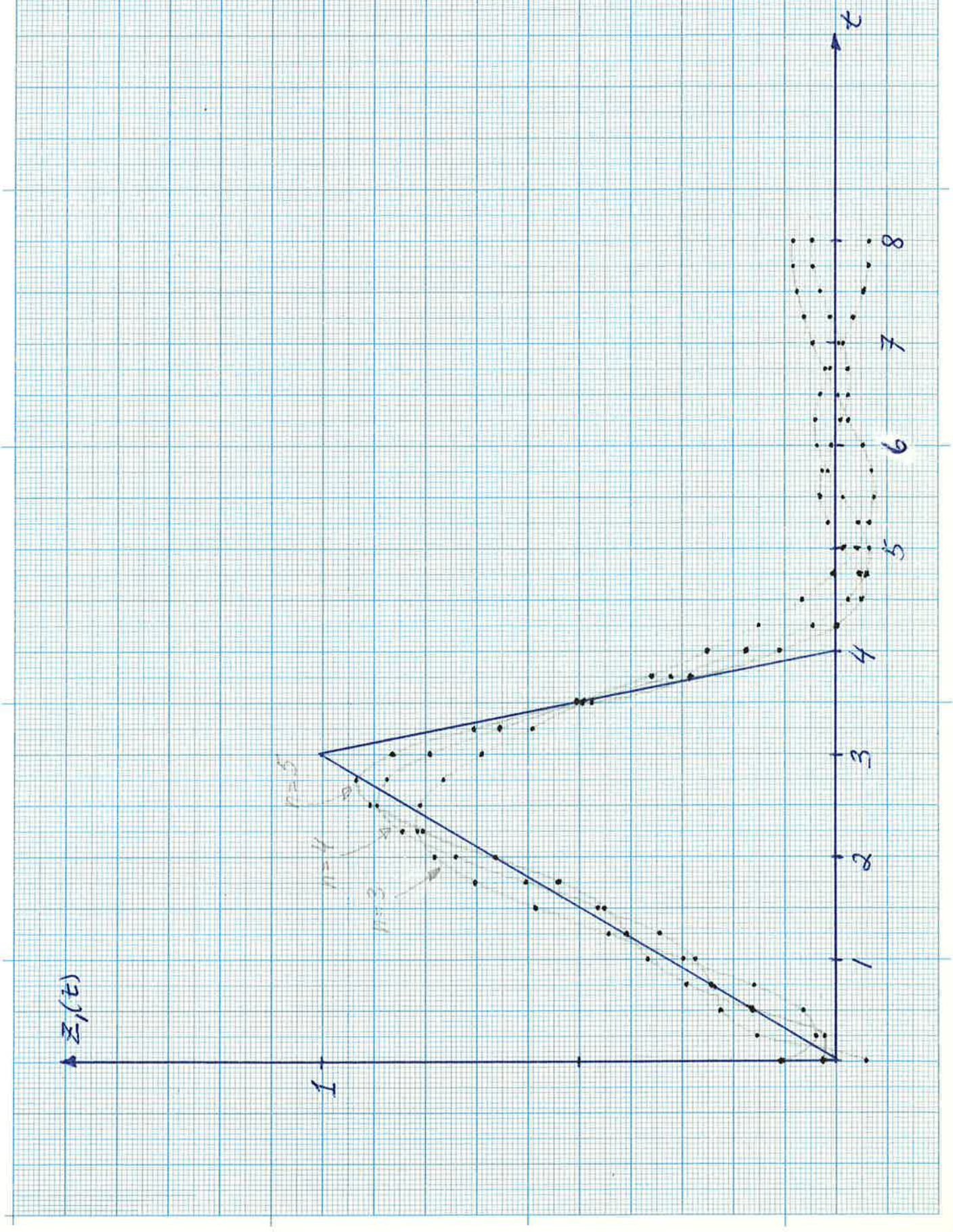
$0.521609 \pm i2.20306$

$8.41823 \pm i8.52246$

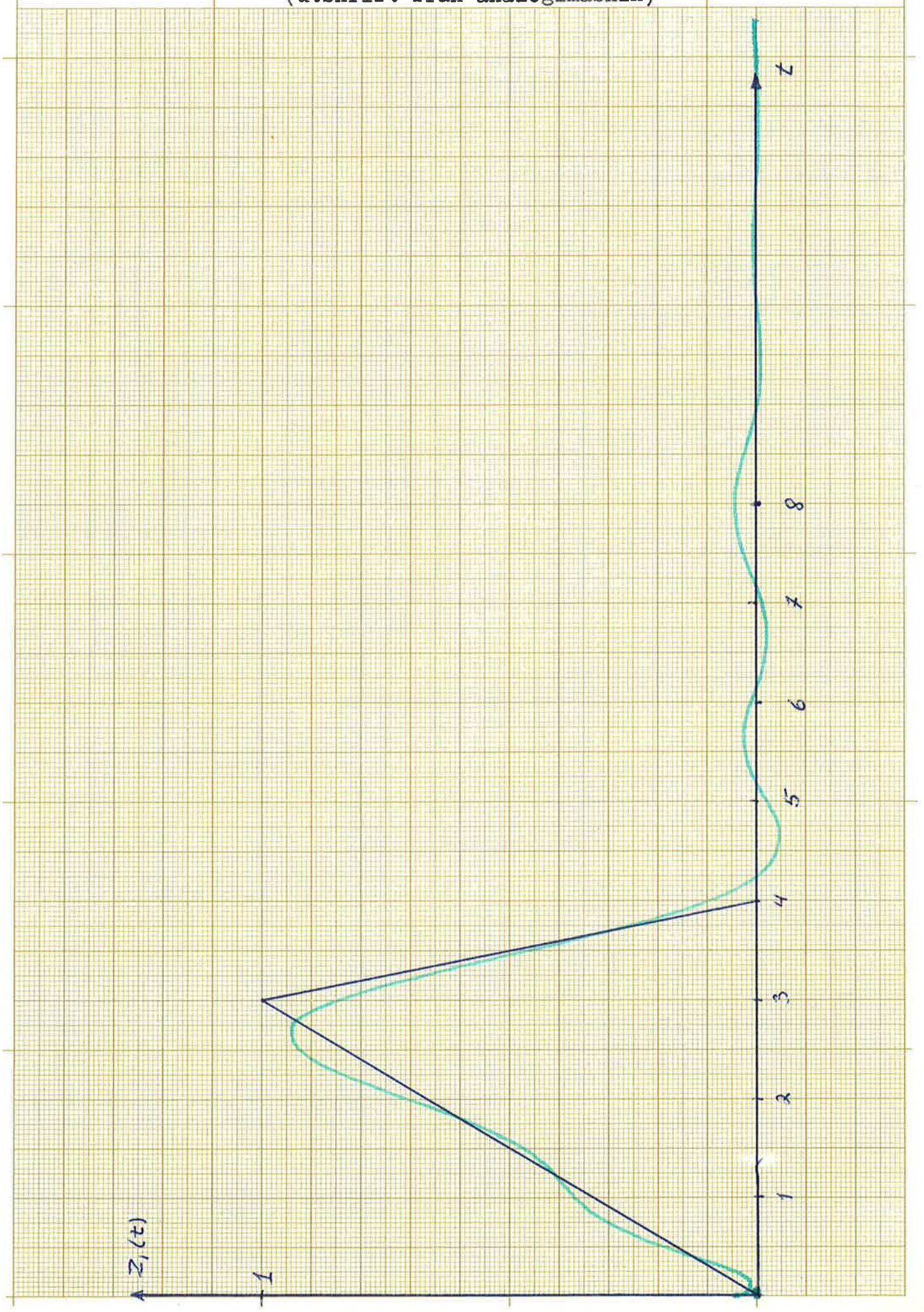
Utsignalen från de samplade systemen ordning n=3,4 och 5 se diagram sida 7:8. Utsignalen från det kontinuerliga systemet ordning n=5, utskrift från analogimaskin, då G(s) köres:se diagram sida 7:9.

De samplade systemens utsignaler

för ordningstalen n=3,4 och 5



Det kontinuerliga systemets utsignal, ordning n=5
(utskrift från analogimaskin)



Referenser

Fletcher-Powells metod:

Computer Journal vol. 6 1963: A rapidly convergent descent method for minimization. (R. Fletcher and M.J.D. Powell)

Newton-Raphsons metod:

1) Non linear mathematics:Saaty and Bram.

2) Walsh: Introduktion to numerical analysis, chapter 8. Minimization of funktions of several variables (M.J.D. Powell).

Lineära minimeringar:

Booth: Numerical Methods.

Signaldetektering:

Kompendium LTH, Teletransmissionsteori allmän kurs II, signaldetektering.

Algolprogram för Fletcher-Powells minimerings-metod

(1/4)

```
START:
begin
    comment Det här programmet ,som utför minimering enligt
    Fletcher-Powells metod består av 4 delar,nämligen huvud,
    procedur M,procedur FP och administrerande program.
    Minimeringen utförs för n=r,r+1,...,1.ym[t] är impulssvaret
    i punkterna t=0,1,...,N-1 Alltså N st. mätpunkter.
    Delta sättes lämpligen till 10-3.Startvärden på a och b väljes
    enligt kap.6;
 integer N,r,n,p,k,i,j,m,g,h,l,s;real V,V1,V2,alfa,alfa2,D,delta;
 real array a,b,c,d,Z,
VA,VB[1:8],grad,SS,
C[1:16],W,H[1:16,1:16],ym[1:40];
;
```

```
procedure M(a,b,V,VA,VB,W,k);value k;integer k;realV;real array a,b,VA,VB,W;
begin comment den här proceduren beräknar V(a,b),gradV ( VA och VB ) samt
andraderivatmatrisen W.Om förlustfunktionen V är större än föregående
förlustfunktion V1 sker hopp till det i ett överordnat block deklarerade
läget PP ;
integer i,j,t,t1; real p,Q,QA,R,S,G;real arrayZ,ZA,ZB,ZA1,ZB1  u[1:11];
     Z[n+1]:=0; for j:=1 step 1 until n do  Z[j]:=b[j];
      V:=(Z[1]-ym[1])↑2;
     for t:=2 step 1 until N do
         begin p:=Z[1]; if (p>₁₀3)∨(-p>₁₀3)then go to PP;
         for i:=1 step 1 until n do Z[i]:=-a[i]×p +Z[i+1];
             Q:=Z[1]-ym[t]; V:=V+Q↑2;
         end;
     if V>V1 then go to PP; if V1=₁₀36 then go to A;
s:=s+1;if s=1 then go to A;if s=2 then begin V2:=V;alfa2:=alfa;go toPP;end;
if V<V2 then begin V2:=V;alfa2:=alfa;go to PP end;
if V>(V1+V2)/2 then begin alfa:=alfa2; go to OPT end;
R:=(V2-V1)/(V-V1);
alfa:=(alfa2↑2-R×alfa↑2)/(2×alfa2-2×R×alfa);
OPT: V:=V1; s:=0; for i:=1 step 1 until n do begin a[i]:=c[i];b[i]:=d[i]; end
  go to P;
A:   ZB1[1]:=ZB[1]:=1; for j:=1step 1 until n do Z[j]:=b[j];
     for j:=1 step 1 until n do begin  VA[j]:=VB[j]:=0;
     ZB[j+1]:=ZB1[j+1]:=ZA[j]:=ZA1[j+1]:=0;
     for i:=j step -1 until 1 do begin W[i,j]:=0; W[n+i,n+j]:=0; end;
     for i:=1 step 1 until n do W[i,n+j]:=0;    end;
     t1:=N-n+1;
     for t:=1 step 1 until N do
begin
     Q:=Z[1]-ym[t]; R:=2×Q;
              for j:=1 step 1 until n do
                  begin VA[j]:=VA[j]+R×ZA[j];
                        VB[j]:=VB[j]+R×ZB[j];
                  end;
          if k=1 then go to L;
              if t>t1 then
              for j:=n-1 step -1 until 1 do
              begin for i:=j step -1 until 1do
                  begin W[i+1,j+1]:=W[i,j]; W[n+i+1,n+j+1]:=W[n+i,n+j]; end;
                  for i:=n-1 step -1 until 1 do W[i+1,n+j+1]:=W[i,n+j];
              end;
              S:=2×ZA[1]; G:=2×ZB[1];
              for j:=1 step 1 until n do
                  begin W[1,j]:=W[1,j]+S×ZA[j]; W[1,n+j]:=W[1,n+j]+S×ZB[j];
                        W[n+1,n+j]:=W[n+1,n+j]+G×ZB[j];
                  end;
              for j:=2 step 1 until n do  W[j,n+1]:=W[j,n+1]+G×ZA[j];
              L:
      p:=Z[1]; for i:=1 step 1 until n do Z[i]:=-a[i]×p+Z[i+1];
  for j:=n-1 step -1 until 1 do
              begin ZA[j+1]:=ZA[j];  ZB[j+1]:=ZB[j]; end;
              for i:=1 step 1 until n do
              begin u[i]:=if i=1 then p else 0;
                    ZA1[i]:=-a[i]×ZA[1]+ZA1[i+1]-u[i];
                    ZB1[i]:=-a[i]×ZB[1]+ZB1[i+1];
                  end;
              ZA[1]:=ZA1[1]; ZB[1]:=ZB1[1];
          end;
     if k=1 then go to B;
     for j:=1 step 1 until 2×n -1 do
     for i:=2×n step -1 until j+1 do
         W[i,j]:=W[j,i];
     B:
end  slut procedur M;;
```

```
procedure FP(a,b,H,grad,p,alfa); value p; integer p; real  alfa;
real array a,b,H,grad;
begin comment Proceduren beräknar ett nytt a och b med hjälp av
    Fletcher-Powells metod  H, grad,C,alfa,p och D förutsättes
    deklarerade i ett över ordnat block.Startvärden på H och grad,alfa,
    a,b erfodras. Proceduren M(a,b,V,VA,VB,W,k) förutsättes
    tillgänglig.;
    real  S,SK1,SK2; integer i,j; real array y[1:12],A[1:12,1:12],
    B[1:12,1:12],z[1:12];
    if alfa>0.9 then begin
    for i:=1 step 1 until p do
begin SS[i]:=0;for j:=1 step 1 until p do SS[i]:=SS[i]+H[i,j]×grad[j];
end;end; for i:=1 step 1 until n do
begin C[i]:=-alfa×SS[i]; C[i+n]:=-alfa×SS[i+n]; a[i]:=a[i]+C[i];
b[i]:=b[i]+C[i+n]end;
 M(a,b,V,VA,VB,W,1); S:=0; for i:=1 step 1 until n do
begin y[i]:=VA[i]- grad[i]; y[i+n]:=VB[i]- grad[i+n];
        S:=S+C[i]×VA[i]+C[n+i]×VB[i]
end; SK1:=0;SK2:=0; forj:=1step 1 until p do begin
z[j]:=0;SK1:=SK1+C[j]×y[j];
 for i:=1 step 1 until p do
begin SK2:=SK2+ y[j]×H[j,i]×y[i];
     z[j]:=z[j]+H[j,i]×y[i];
end
end;
if SK1<0 then begin punch(1);write(‹SK1=›);print(0,5,SK1);
space(6);write(‹S=›);print(0,5,S);punch(1);s:=0;alfa:=4×alfa;go to PPend;
 for j:=1 step 1 until p do for i:=j step 1 until p do
begin A[j,i]:=C[j]×C[i]/SK1;
    B[j i]:=-z[j]×z[i]/SK2;
    H[j,i]:=H[j,i]+A[j,i]+B[j i]; H[i,j]:=H[j,i]
end;
 for i:=1 step 1  until n do
begin grad[i]:=VA[i]; grad[i+n]:=VB[i]
end;
 D:=0; for i:=1 step 1 until p do D:=D+C[i]↑2; D:=sqrt(D);

end procedur FP;;
```

```
l:=read; N:=read; for i:=1 step 1 until N do ym[i]:=read;
delta:=read;r:=read;
for n:=r step 1 until 1 do
begin for i:=1 step 1 until n do begin a[i]:=read;b[i]:=read; end;
p:=2×n; alfa:=s:=1; write(‹MINIMERING ,MED,FLETCHER,POWELLS,METOD›);
punch(1); write(‹N=›); print(1,0,n); punch(1);

for i:=1 step 1 until p do
for j:= 1 step 1 until p do H[i,j]:= if i=j then 1 else 0;
V1:=10^36;
M(a,b,V,VA VB,W,1); write(‹STARTVÄRDEN›);punch(1);
write(‹V=›);print(0,10 V);punch(1);punch(1);
    for i:=1 step 1 until n do
begin write(‹a=›); print(2,6,a[i]); space(1);
        write(‹B=›); print(b[i]); punch(1);punch(1);
end;
for i:=1 step 1 until n do begin grad[i]:=VA[i]; grad[i+n]:=VB[i]; end;
L:  V1:=V;
    for i:=1 step 1 until n do  begin c[i]:=a[i]; d[i]:=b[i]; end;
P:  FP(a,b,H,grad,p,alfa);
punch(1);
write(‹V=›); print(0,10,V); punch(1);punch(1);
    for i:=1 step 1 until n do
begin write(‹a=›); print(2,6,a[i]); space(1);
        write(‹B=›); print(b[i]); punch(1);punch(1);
end;
if(D/alfa<delta)∧(D<delta) then begin write(‹S,ÄR,MINDRE,ÄN,DELTA›);
go to UT end;
alfa:=s:=1; go to L;
PP: for i:=1 step 1 until n do begin a[i]:=c[i]; b[i]:=d[i] end;
alfa:=alfa/2; go to P;
UT:   for j:=1 step 1 until p do
begin for i:=1 step 1 until p do
    begin print(5,6,H[j,i]); space(1); end; punch(1);
end; write(‹D=›);print(0,5,D); space(5); write(‹ ALFA=›); print(alfa);
 punch(1);write(‹C=›); space(1); for i:=1 step 1 until p do
begin print(C[i]); space(1);
end; write(‹GRAD=›);space(1);
for i:=1 step 1 until p do
begin print(grad[i]); punch(0);end;
punch(1); Z[n+1]:=0;for j:=1 step 1 until n do Z[j]:=b[j];
for i:=2 step 1  until N do
begin V:=Z[1]; for j:=1 step 1 until n do
        Z[j]:=-a[j]×V+Z[j+1]; print(2,0,i);space(5);print(2,6,Z[1]);
        punch(1);punch(1);
end;
punch(15);
end loop n ;
go to START
end
```

```
START:
begin comment Programmet minimerar en funkti n V(a,b), där
a=<a.1,     a.n> och b=<b 1,  ..b n> med Newton-Raphsons metod.
I programmet (mellan huvudprogrammets båda delar) skall procedurerna
M, invers och NR införas.Kontrollera fältet på ym (antalet mätdata),
N antalet mätdata(Obs  N=1 motsvarart=0),delta är testkvantitetens
högsta värde. System av ordning r t.o m  l beräknas;
    integer N,p,n,k,i,j,m,l,s,r; real V,V1,V2,eps,D,alfa,alfa2,delta;
    real array a,b,c,d,Z,VA,VB[1:10],WW,W[1:20,1:20],C[1:20],ym[1:40];  ;
```

```
procedure M(a,b,V,VA,VB,W,k);value k;integer k;realV;real array a,b,VA,VB,W;
begin comment den här proceduren beräknar V(a,b),gradV ( VA och VB ) samt
andraderivatmatrisen W.Om förlustfunktionen V är större än föregående
förlustfunktion V1 sker hopp till det i ett överordnat block deklarerade
läget PP ;
integer i,j,t,t1; real p,Q,QA,R,S,G;real arrayZ,ZA,ZB,ZA1,ZB1 u[1:11];
     Z[n+1]:=0; for j:=1 step 1 until n do  Z[j]:=b[j];
      V:=(Z[1]-ym[1])↑2;
     for t:=2 step 1 until N do
         begin p:=Z[1]; if (p>10 3)∨(-p>10 3)then go to PP;
         for i:=1 step 1 until n do Z[i]:=-a[i]×p +Z[i+1];
         Q:=Z[1]-ym[t]; V:=V+Q↑2;
         end;
     if V>V1 then go to PP; if V1=10 36 then go to A;
s:=s+1;if s=1 then go to A;if s=2 then begin V2:=V;alfa2:=alfa;go toPP;end;
 f V<V2 then begin V2:=V;alfa2:=alfa;go to PP end;
if V>(V1+V2)/2 then begin alfa:=alfa2; go to OPT end;
R:=(V2-V1)/(V-V1);
alfa:=(alfa2↑2-R×alfa↑2)/(2×alfa2-2×R×alfa);
OPT: s:=0; for i:=1 step 1 until n do begin a[i]:=c[i];b[i]:=d[i]; end;
 go to P;
;
```

```
A:  ZB1[1]:=ZB[1]:=1; for j:=1step 1 until n do Z[j]:=b[j];
    for j:=1 step 1 until n do begin VA[j]:=VB[j]:=0;
    ZB[j+1]:=ZB1[j+1]:=ZA[j]:=ZA1[j+1]:=0;
    for i:=j step -1 until 1 do begin W[i,j]:=0; W[n+i,n+j]:=0; end;
    for i:=1 step 1 until n do W[i,n+j]:=0;   end;
    t1:=N-n+1;
    for t:=1 step 1 until N do
begin
    Q:=Z[1]-ym[t]; R:=2×Q;
          for j:=1 step 1 until n do
              begin VA[j]:=VA[j]+R×ZA[j];
                    VB[j]:=VB[j]+R×ZB[j];
              end;
        if k=1 then go to L;
          if t>t1 then
          for j:=n-1 step -1 until 1 do
          begin for i:=j step -1 until 1do
              begin W[i+1,j+1]:=W[i,j]; W[n+i+1,n+j+1]:=W[n+i,n+j]; end;
              for i:=n-1 step -1 until 1 do W[i+1,n+j+1]:=W[i,n+j];
          end;
          S:=2×ZA[1]; G:=2×ZB[1];
          for j:=1 step 1 until n do
              begin W[1,j]:=W[1,j]+S×ZA[j]; W[1,n+j]:=W[1,n+j]+S×ZB[j];
                    W[n+1,n+j]:=W[n+1,n+j]+G×ZB[j];
              end;
          for j:=2 step 1 until n do  W[j,n+1]:=W[j,n+1]+G×ZA[j];
          L:
    p:=Z[1]; for i:=1 step 1 until n do Z[i]:=-a[i]×p+Z[i+1];
 for j:=n-1 step -1 until 1 do
          begin ZA[j+1]:=ZA[j];  ZB[j+1]:=ZB[j]; end;
          for i:=1 step 1 until n do
              begin u[i]:=if i=1 then p else 0;
                  ZA1[i]:=-a[i]×ZA[1]+ZA1[i+1]-u[i];
                  ZB1[i]:=-a[i]×ZB[1]+ZB1[i+1];
              end;
          ZA[1]:=ZA1[1]; ZB[1]:=ZB1[1];
        end;
    if k=1 then go to B;
    for j:=1 step 1 until 2×n -1 do
    for i:=2×n step -1 until j+1 do
        W[i,j]:=W[j,i];
    B:
end  slut procedur M;;
```

```
procedure invers(a,n,eps,singular); value n,eps; array a;
integer n; real eps; label singular;
comment Inverterar matrisen a av ordning n med Gauss-Jordans metod.
Matrisen a blir förstörd då inversen bildas utan extra fält. Vid
varje steg användes det abs. största elementet som pivoelement.
Index för succesiva pivoelement placeras i vektorerna r och c,
vilka sedan användes för återpermutering. Om något pivoelement
är mindre än eps går proceduren till läge singular i huvudprogrammet;
begin integer i,j,k,l,pivi,pivj,p; real pivot; integer array r,c[1:30];
    for i:=1 step 1 until n do r[i]:=c[i]:=i;
    comment sök startvärde för pivot; pivi:=pivj:=1;
    for i:=1 step 1 until n do for j:=1 step 1 until n do
    if abs(a[i,j])>abs(a[pivi,pivj]) then
    begin pivi:=i; pivj:=j end;
    comment start lösning;
    for i:=1 step 1 until n do
    begin l:=r[i]; r[i]:=r[pivi]; r[pivi]:=l; l:=c[i];c[i]:=c[pivj];
     c[pivj]:=l;
        if eps>abs(a[r[i],c[i]]) then go to singular;
        for j:=n step -1 until i+1,i-1 step -1 until 1 do
        a[r[i],c[j]]:=a[r[i],c[j]]/a[r[i],c[i]];
        a[r[i],c[i]]:=1/a[r[i],c[i]];
        pivot:=0;
        for k:=1 step 1 until i-1,i+1 step 1 until n do
        begin for j:=n step -1 until i+1,i-1 step -1 until 1 do
            begin  a[r[k],c[j]]:=a[r[k],c[j]]-a[r[i],c[j]]×a[r[k],c[i]];
                if k>i∧j>i∧abs(a[r[k],c[j]])>abs(pivot) then
                begin pivi:=k; pivj:=j; pivot:=a[r[k],c[j]] end test
            end j loop;
            a[r[k],c[i]]:=-a[r[i],c[i]]×a[r[k],c[i]];
        end k loop;
    end i loop och lösning;
    comment start återpermutation av rader för z=1 och av kolonner för z=2;
    begin  integer array tag, loc[1:30]; integer z,i,t; real w;
        for z:=1,2 do
        begin for i:=1 step 1 until n do tag[i]:=loc[i]:=i;
            for i:=1 step 1 until n do
            begin if z=1 then
                begin t:=r[i]; j:=loc[t]; k:=c[i] end
                else begin t:=c[i]; j:=loc[t]; k:=r[i] end;
                if j≠k then
                begin for p:=1 step 1 until n do
                    begin if z=1 then
                        begin w:=a[j,p]; a[j,p]:=a[k,p]; a[k,p]:=w end
                        else begin w:=a[p,j]; a[p,j]:=a[p,k]; a[p,k]:=w end
                    end p loop;
                    tag[j]:=tag[k]; tag[k]:=t;
                    loc[t]:=loc[tag[j]]; loc[tag[j]]:=j
                end j,k test
            end i loop
        end z loop
    end permutation
end invers;;;
```

```
procedure NR(a,b,p,alfa); value p;integer p; real alfa;
realarray a,b;
begin comment proceduren beräknar ett nytt a och b med Newton-
raphsons metod,samt därtill hörande V,VA,VB,W. Procedurerna invers
och M förutsättes införda i huvudprogrammet  Fälten C,VA,VB,W och
real V,V1,eps,D och läge singular deklarerade i ett överordnat block;
    integer i,j; real g,t;
    if alfa=1 then
begin for i:=1 step 1 until p do C[i]:=0;
    for j:=1step 1 until p do
    for i:=1 step 1 until n do
begin g:=W[j,i]×VA[i];t:=W[j,i+n]×VB[i];
    C[j]:=C[j]-alfa×(g+t)
end
end;
    for i:=1 step 1 until n do
begin a[i]:=a[i]+C[i];b[i]:=b[i]+C[i+n]
end;
    M(a,b,V,VA,VB,W,0);
    D:=0; for i:=1 step 1 until p do
    D:=D+C[i]↑2;
    D:=sqrt(D)
end procedure NR;;
```

```
l:=read; N:=read; for i:=1 step 1 until N do ym[i]:=read;
    delta:=read;r:=read; eps:=10-8; for n:=r step 1 until 1 do
begin alfa:=s:=1; for i:=1 step 1 until n do
begin a[i]:=read; b[i]:=read
end;  p:=2×n; m:=1; V1:=1036;
    M(a,b,V,VA,VB W,0); write(‹STARTVÄRDEN,NEWTON/RAPHSONS,METOD›);
    punch(0);   write(‹N=›); print(1,0,n);punch(1);
   for i:=1 step 1 until n do
begin write(‹A=›);print(0,3,a[i]);punch(0);write(‹B=›);print(0,3,b[i]);
    punch(1)
end; write(‹V=›);print(0,5,V); punch(1);
    L: V1:=V;for i:=1 step 1 until n do
begin c[i]:=a[i]; d[i]:=b[i]
end;  invers(W,p,eps,singular);
    P:if alfa<10-10 then begin punch(1);write(‹ALFA,FÖR,LITET›);
    punch(1); go to singular end;
    NR(a,b,p,alfa);
    if m=6  then
begin punch(15);m:=1
end; m:=m+1;
for i:=1 step 1 until n do
begin write(‹a=›); print(0,5,a[i]); punch(0);
      write(‹b=›); print(0,5,b[i]); punch(1)
end;
   write(‹V=›); print(0,5,V); punch(1);
write(‹D=›);print(0,5,D); punch(1);
    write(‹alfa=›); print(1,5,alfa);punch(1);
    if ((D/alfa)<delta)∧(D<delta) then begin write(‹S,MINDRE,
    ÄN,DELTA›); go to singular end;
    alfa:=s:=1; go to L;
    PP:for i:=1 step 1 until n do
begin a[i]:=c[i]; b[i]:=d[i]
end;  alfa:=alfa/2;go to P;
singular: punch(1); write(‹SLUTLIGA,INVERSEN›);punch(1);
     invers(W,p,eps,singular);
        for i:=1 step 1 until p do begin  for j:=1 step 1 until  p do
    begin print(0,5,W[i,j]); punch(0) end; punch(1) end; punch(1);
Z[n+1]:=0; for j:=1 step 1 until n do Z[j]:=b[j];
    for j:=2 step 1 until N do begin punch(0); print(3,0,j); punch(0);
    V:=Z[1]; for i:=1step 1 until n do Z[i]:=-a[i]×V+Z[i+1];
    print(0,5,Z[1]) end; punch(15)
end; go to START
end
```