NUMERISK LÖSNING AV OPTIMALA STYR-PROBLEN VIA GENERALISERADE MULTIPLI-KATORFUNKTIONER

HANS ANDERS JOHANSSON

RE-153 Februari 1975
Inst. för Reglerteknik
Lunds Tekniska Högskola

Numerisk lösning av optimala styrproblem via generaliserade multiplikatorfunktioner.

Hans Anders Johansson

RE-153 februari 1975
Institutionen för Reglerteknik
Lunds Tekniska Högskola

NUMERISK LÖSNING AV OPTIMALA STYRPROBLEM VIA GENERALISERADE MULTIPLIKATORFUNKTIONER.

Examensarbete vid Institutionen för Reglerteknik,
Lunds tekniska högskola.
Utfört under vår- och höstterminerna 1974 av Hans A Johansson.
Handledare: Forskarassistent Krister Mårtensson och
Civilingenjör Torkel Glad.

1. ABSTRACT

In this paper a computerprogram IMF1 for numerical solution of optimal control-problems is presented.
The method is based on a generalizing of the multiplierfunction idea into infinite-dimensional problems.
The program is written in FORTRAN and to compute different examples only a change of a subroutine USER is needed.
The program has been tested on problems of varying degree of difficulty and it has been compared to other solutionmethods.

I detta examensarbete presenteras ett datorprogram IMF1 för numerisk lösning av optimala styrproblem.
Programmet är baserat på en generalisering av multiplikator-funktionsbegreppet till oändligt-dimensionella problem.
Programmet är skrivet i FORTRAN och för att köra olika exempel på dator behöver endast en subrutin USER ändras.
Programmet har testats på problem av varierande svårighetsgrad och jämförelser har gjorts med andra lösningsmetoder.

2. Innehållsförteckning.

3. Problemformulering.

Givet: Ett dynamiskt system beskrivet av

$\dot{x}=f(x(t),u(t),t)$ där

$t_0 \leq t \leq t_1$, $\quad x=\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \cdot \\ \cdot \\ x_n(t) \end{bmatrix}$ har dimensionen n,

$f=\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \cdot \\ \cdot \\ f_n \end{bmatrix}$ och $u=\begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ \cdot \\ \cdot \\ u_n(t) \end{bmatrix}$

t_0 och t_1 är start resp. sluttidpunkt.

Randvärde: $x(t_0)=x_0$

Sökt: De trajektorier x(t) och u(t) som minimerar

$$H(x(t),u(t))=F(x(t_1))+\int_{t_2}^{t_1} L(x(t),u(t))dt$$

under det att bivillkoren $\dot{x}(t)=f(x(t),u(t))$ och

$x(t_0)=x_0$ är uppfyllda.

Lösningsmetoden går ut på att man minimerar

$$H(\lambda,\dot{x}(t),u(t)) = F(x(t_1)) + \int_{t_0}^{t_1} L(x,u)\,dt +$$

$$+ \langle \lambda, f-\dot{x} \rangle + c\langle f-\dot{x}, f-\dot{x} \rangle$$

$$\lambda = \begin{bmatrix} \lambda_1(t) \\ \lambda_2(t) \\ \lambda_3(t) \\ \vdots \\ \lambda_n(t) \end{bmatrix}$$

Fördelen med detta är att man slipper lösa differentialekvationen $\dot{x}=f(x(t),u(t),t)$.
Det finns ett tal c_0 så att om $c>c_0$ och $\lambda \equiv \lambda^*$ är de optimala multiplikatorfunktionerna så har $H(\lambda^*,u)$ ett lokalt minimum för $u=u^*$ där u^* ger minimum för H.
För närmare studium se referens (3).
x_i är tillståndsvariablerna och u_i styrvariablerna. Observera att dessa är funktioner av tiden.
F är en funktion som endast beror av sluttillståndet för x.
c är en konstant.
$\langle \lambda, f-\dot{x} \rangle$ är en skalärprodukt och beräknas enligt:

$$\int_{t_0}^{t_1} \lambda^T (f-\dot{x})\,dt.$$

4. Teori.

Lösningsmetoden baseras på Fletcher-Reeves konjugerade gradientmetod. Flödesschema för denna finns i Appendix A. Fletcher-Reeves metod innebär att man söker minimum längs en riktning, beräknar gradienten i minimumpunkten och med hjälp därav beräknar en ny sökriktning. Referens (4).

Algoritm för F-R metod:

1. Beräkna gradienten g_0 för givet x_0
$g_0=\nabla f(x_0)$ och sätt begynnelsesökriktningen
$d_0=-g_0$.

2. För k=0,1,....,n-1

a) $x_{k+1}=x_k+\alpha_k d_k$ där α_k minimerar $f(x_k+\alpha d_k)$.

b) Beräkna nya gradienten $g_{k+1}=\nabla f(x_{k+1})$.

c) Om k=n-1 sätt nya sökriktningen $d_{k+1}= -g_{k+1}+\beta d_k$
där $$\beta_k = -\frac{\langle g_{k+1}, g_{k+1}\rangle}{\langle g_k, g_k\rangle}$$

3. Sätt $x_0=x_n$ och börja om på 1.

Denna algoritm gäller för det n-dimensionella fallet, för rent kvadratiska problem. Om problemet inte är kvadratiskt behövs mer än n st sökriktningar.
För mitt problem måste algoritmen modifieras:

punkt 2c blir istället:

Om $\int_{t_0}^{t_1} g_{k+1}^T g_{k+1} dt > \varepsilon$, ε är ett konstant tal,
så sätt den nya sökriktningen $d_{k+1}= -g_{k+1}+\beta d_k$
där $$\beta_k = \frac{\int_{t_0}^{t_1} g_{k+1}^T g_{k+1} dt}{\int_{t_0}^{t_1} g_k^T g_k dt}$$

och punkt 3 blir: $\lambda_{k+1}= \lambda_k + 2c(f-\dot{x})$,
sätt $x_0=x_n$ och börja om på 1.

Förklaring av och kommentar till algoritmens flödesschema (se Appendix A):

I subrutinen USER finns de gissade begynnelsevärdena ξ_0 och λ_0. ξ_0 är sammansatt av x_0 och u_0.
Gradienten g_0 av funktionen H map ξ beräknas.
Första sökriktningen sätts lika med negativa gradienten. Därefter minimeras H längs sökriktningen vilket ger α_k.
ξ_{k+1} sätts till det värde som ξ har i det funna minimat. Gradienten i denna punkt beräknas, varefter kvoten mellan skalärprodukterna av gamla och nya gradienterna med sig själva beräknas. Denna används sedan för att beräkna nya sökriktningen d_{k+1}.
Om skalärprodukten av nya gradienten med sig själv är tillräckligt liten beräknas λ_{k+1} varvid $f-\dot{x}$ adderas till λ_k. $f-\dot{x}$ är ett mått på hur väl differentialekvationen $\dot{x}=f$ är uppfylld. Är skillnaden tillräckligt liten för alla $\dot{x}_i(t)$ vid varje tidpunkt stoppas programmet.
Som synes är det av grundläggande betydelse att gradienten av H map ξ går att beräkna.
Denna härledes därför i nästa avsnitt.

Härledning av gradienten av H.

a) map u:

$$H(\dot{x},u)=F(x(t_1))+\int_{t_0}^{t_1}L(x,u)dt+\langle\lambda,f(x,u)-\dot{x}\rangle+$$

$$+ c\langle f(x,u)-\dot{x},f(x,u)-\dot{x}\rangle$$

Vi definierar differentialen av H map u med inkrementet h som:

$$\delta H_u(\dot{x},u,h)=\lim_{\varepsilon\to 0}\frac{H(\dot{x},u+\varepsilon h)-H(\dot{x},u)}{\varepsilon}$$

om $\delta H(\dot{x},u,h)$ existerar och är linjär och kontinuerlig i h.
Låt oss beräkna denna differential:

$$H(\dot{x},u+\varepsilon h)=\int_{t_0}^{t_1}L(x,u+\varepsilon h)dt+\langle\lambda,f(x,u+\varepsilon h)-\dot{x}\rangle+$$

$$+c\langle f(x,u+\varepsilon h)-\dot{x},f(x,u+\varepsilon h-\dot{x}\rangle$$

Taylorutveckla L och f kring u.

$$H(\dot{x},u+\varepsilon h)=\int_{t_0}^{t_1}(L+L_u\varepsilon h+o(\varepsilon)dt+\langle\lambda,f+f_u\varepsilon h+O(\varepsilon)-\dot{x}\rangle+$$

$$+\langle f+f_u\varepsilon h+O(\varepsilon)-\dot{x},f-f_u\varepsilon h+O(\varepsilon)-\dot{x}\rangle$$

Vi får då :

$$\frac{H(\dot{x},u+\varepsilon h)-H(\dot{x},u)}{\varepsilon}=\int_{t_0}^{t_1}(L_uh+\frac{O(\varepsilon)}{\varepsilon})dt+$$

$$+\langle\lambda,f_uh+\frac{O(\varepsilon)}{\varepsilon}\rangle+2c\langle f,f_uh\rangle-2c\langle\dot{x},f_uh\rangle+\frac{O(\varepsilon)}{\varepsilon}$$

Låt $\varepsilon \to 0$. Då fås differentialen

$$(1) \qquad \delta H_u(\dot{x},u,h) = \int_{t_0}^{t_1} L_u h\,dt + \langle \lambda, f_u h \rangle + 2c\langle f-\dot{x}, f_u h \rangle$$

Observera att vi har definierat skalärprodukten som

$$\langle x,y \rangle = \int_{t_0}^{t_1} x^T y\,dt$$

Vi kan då skriva (1) som

$$(2) \qquad \delta H_u(\dot{x},u,h) = \langle (L_u^T + f_u^T\lambda + 2cf_u^T(f-\dot{x})), h \rangle .$$

Hur bestämmes nu gradienten? Jo om

$\delta H_u(\dot{x},u,h)$ kan skrivas som

$$\delta H_u = \langle \nabla_u H, h \rangle$$

så defenierar vi gradienten av H map u som $\nabla_u H$. Men (2) är ju precis på denna formen. Vi har alltså

$$\nabla_u H = L_u^T + f_u^T\lambda + 2cf_u^T(f-\dot{x})$$

b) map $\dot{x}$

Detta är lite knepigare eftersom vi måste komma ihåg att $\dot{x}$ bestäms ur x genom

$$x(t)=x_0+\int_{t_0}^{t_1}\dot{x}dt$$

Ger vi alltså $\dot{x}(t)$ ett tillskott $\dot{h}(t)$ kommer x(t) att få tillskottet $\int\dot{h}(s)ds=h(t)-h(t_0)=h(t)$ eftersom $h(t_0)=0$, dvs $x(t_0)$ fix.
Vi kan då beräkna differentialen av H map $\dot{x}$ med inkrementet $\dot{h}$ som

$$\delta H_{\dot{x}}(\dot{x},u,\dot{h})=\lim_{\varepsilon\to 0}\frac{H(\dot{x}+\varepsilon\dot{h},u)-H(\dot{x},u)}{\varepsilon}$$

om $\delta H_{\dot{x}}$ existerar och är linjär och kontinuerlig i h. Vi har

$$H(\dot{x}+\varepsilon\dot{h},u)=F(x(t_1)+\varepsilon h(t_1))+\int_{t_0}^{t_1}L(x+\varepsilon h,u)dt+$$

$$+<\lambda,f(x+\varepsilon h,u)-\dot{x}-\varepsilon\dot{h}>+<f(x+\varepsilon h,u)-\dot{x}-\varepsilon\dot{h},f(x+\varepsilon h,u)-\dot{x}-\varepsilon\dot{h}>$$

Taylorutveckla L och f och F kring x och skriv $<\xi,\eta>$ som $\int\xi^T\eta dt$

$$H(\dot{x}+\varepsilon\dot{h},u)=F(x(t_1))+F_x(x(t_1))h(t_1)\ \varepsilon+\emptyset(\varepsilon)+$$

$$+\int_{t_0}^{t_1}(L+L_x\varepsilon h+O(\varepsilon)+\lambda^Tf+\lambda^Tf_x\varepsilon h+O(\varepsilon)-\lambda^T\dot{x}-\lambda^T\varepsilon\dot{h}+$$

$$+c(f+f_x\varepsilon h+O(\varepsilon)-\dot{x}-\varepsilon\dot{h})^T(f+f_x\varepsilon h+O(\varepsilon)-\dot{x}-\varepsilon\dot{h}))dt=$$

$$=F(x(t_1))+\int_{t_0}^{t_1}(F_x(x(t_1))\dot{h}\varepsilon+L+L_x\varepsilon h+\lambda^T(f-\dot{x})+\lambda^Tf_x\varepsilon h-\dot{\lambda}^T\varepsilon\dot{h}+$$

$$+c(f-\dot{x})^T(f-\dot{x})+2c(f^Tf_x\varepsilon h-\dot{x}^Tf_x\varepsilon h-f^T\varepsilon\dot{h}+x^T\varepsilon\dot{h})+O(\varepsilon))dt$$

Efter lite räkningar finner man då att

$$(3)\qquad \delta H_{\dot{x}}(\dot{x},u,\dot{h}) = \int_{t_0}^{t_1}(L_x+\lambda^T f_x+2c(f-\dot{x})^T f_x)h\,dt -$$
$$- \int_{t_0}^{t_1}(\lambda^T+2c(f-\dot{x})^T-F_X(x(t_1)))\dot{h}\,dt$$

Den första termen skriver vi om med hjälp av följande lemma.

Lemma: Om $h(t)$ deriverbar och $h(t_0)=0$

så $\int_{t_0}^{t_1}\xi^T(t)h(t)dt = \int_{t_0}^{t_1}(\int_{t}^{t_1}\xi^T(s)ds)\dot{h}(t)dt$

för alla $\xi(t)$.

Alltså kan första termen i (3) skrivas om som

$$\int_{t_0}^{t_1}(L_x+\lambda^T f_x+2c(f-\dot{x})^T f_x)h\,dt =$$
$$= \int_{t_0}^{t_1}(\int_{t_0}^{t_1}(L_x+\lambda^T f_x+2c(f-\dot{x})^T f_x)ds)\dot{h}(t)dt$$

och vi får

$$\delta H_{\dot{x}}(\dot{x},u,\dot{h}) = (\int_{t_0}^{t_1}F_x^T(x(t_1))-\lambda^T-2c(f-\dot{x})^T+$$
$$+\int_{t_0}^{t_1}(L_x+\lambda^T f_x+2c(f-\dot{x})^T f_x)ds)\dot{h}\,dt$$

Alltså kan $\delta H_{\dot{x}}$ skrivas som

$$\delta H_{\dot{x}} = <\nabla_{\dot{x}}H, h>$$

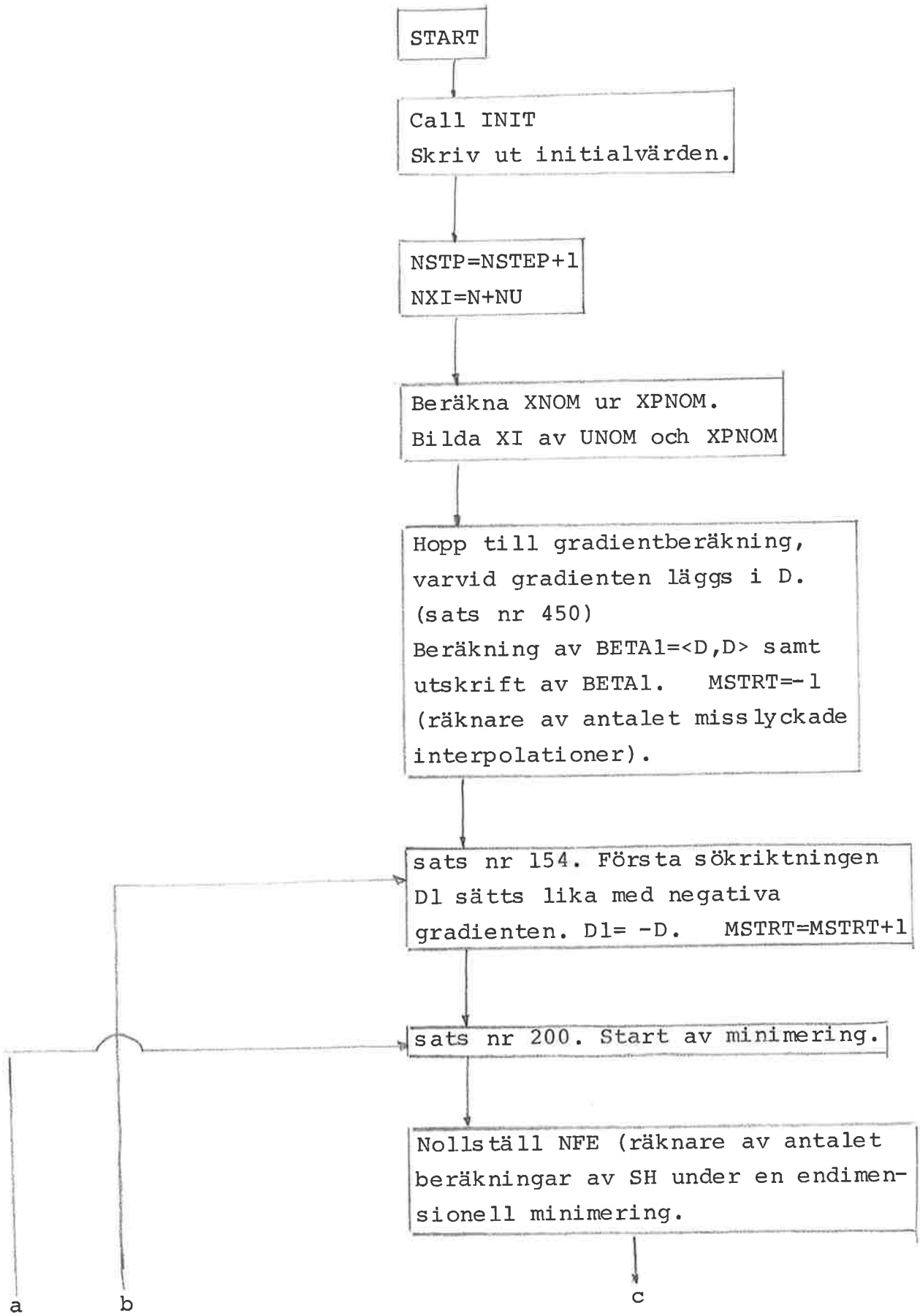
och gradienten $\nabla_{\dot{x}}H$ ges alltså av

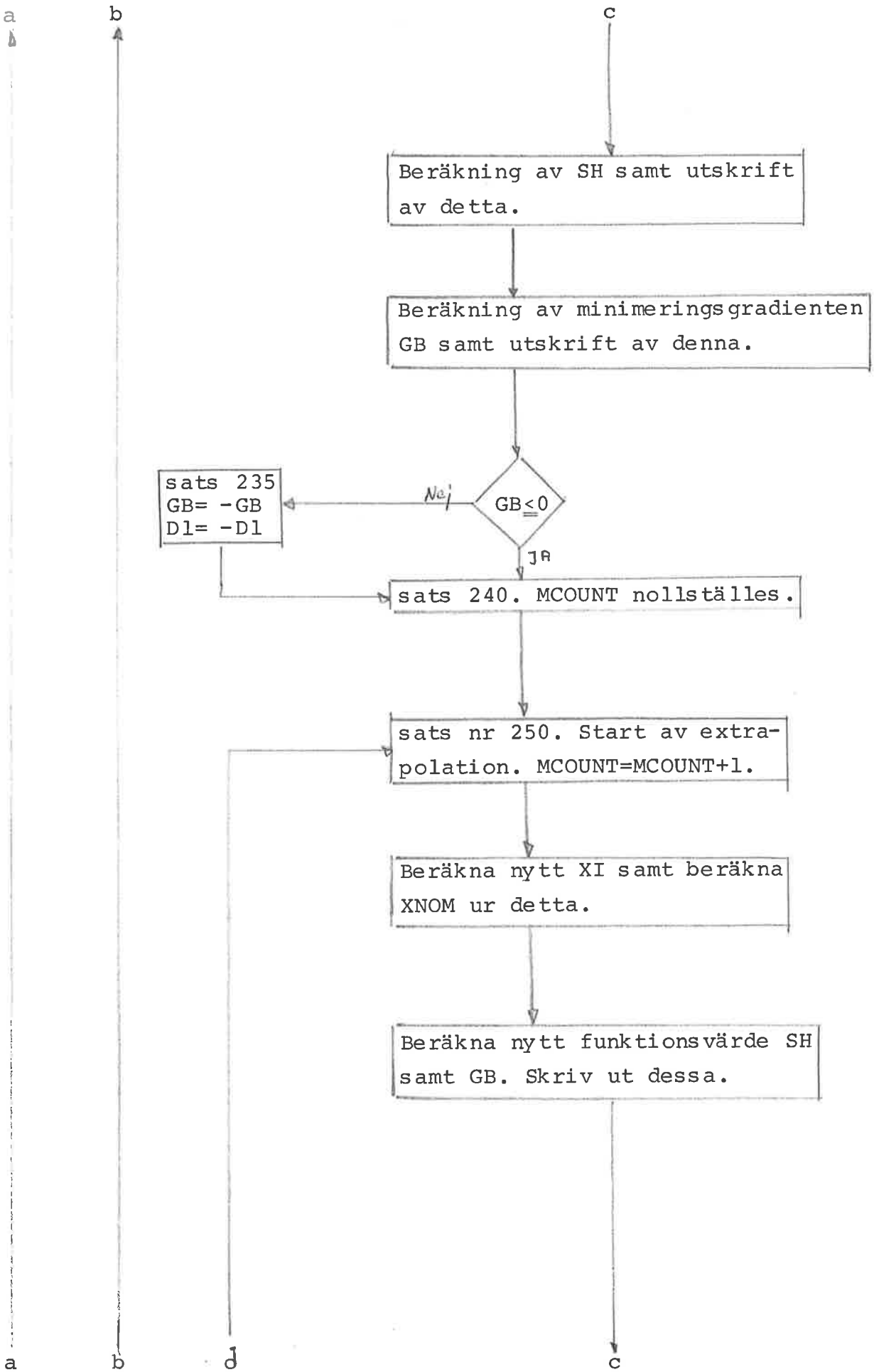
$$\nabla_{\dot{x}}H = F_x^T(x(t_1))-\lambda-2c(f-\dot{x})+\int_{t}^{t_1}(L_x^T+f_x^T\lambda+2cf_x^T(f-\dot{x})ds$$

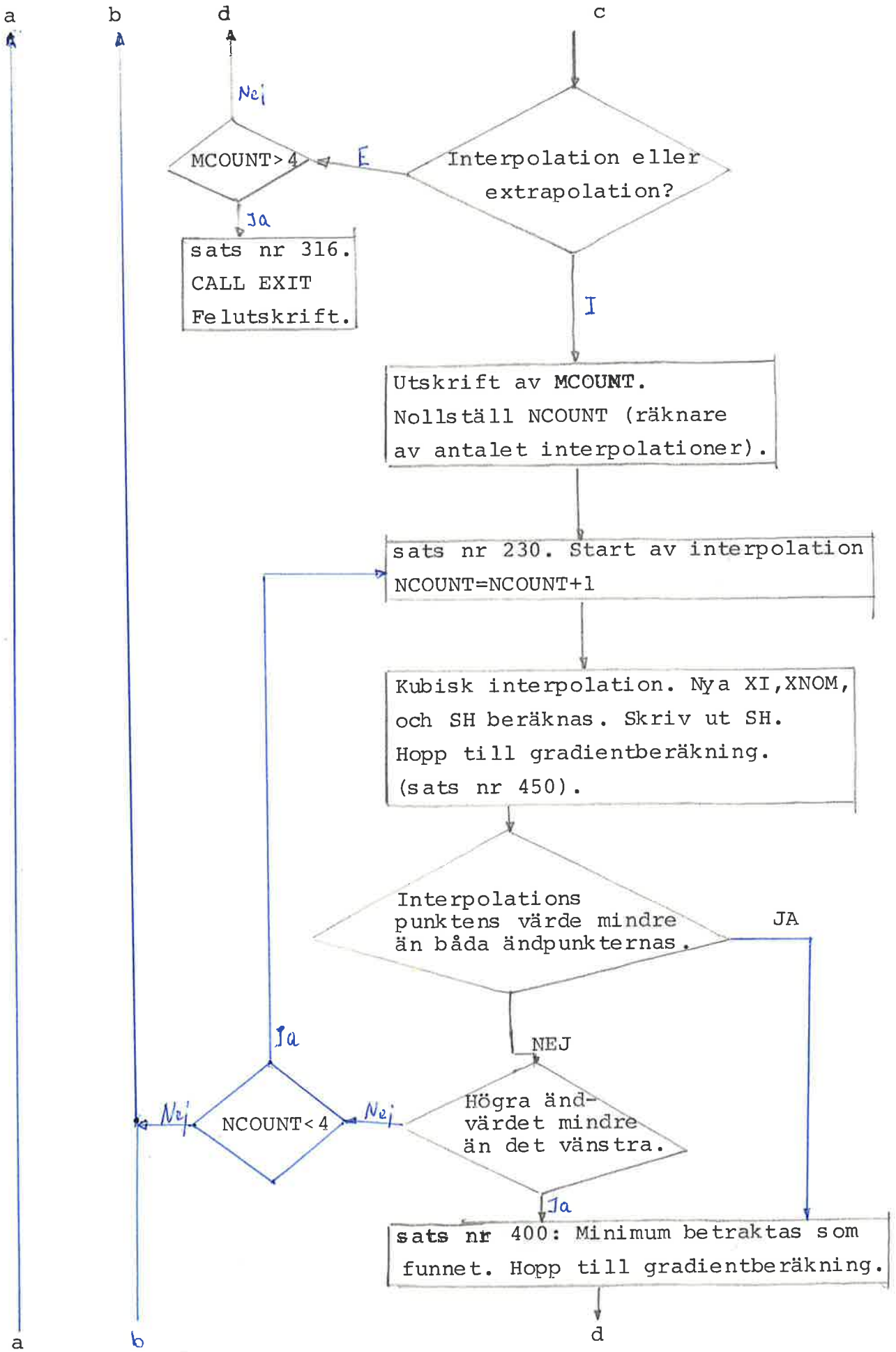
5. Flödesschema för programmet.

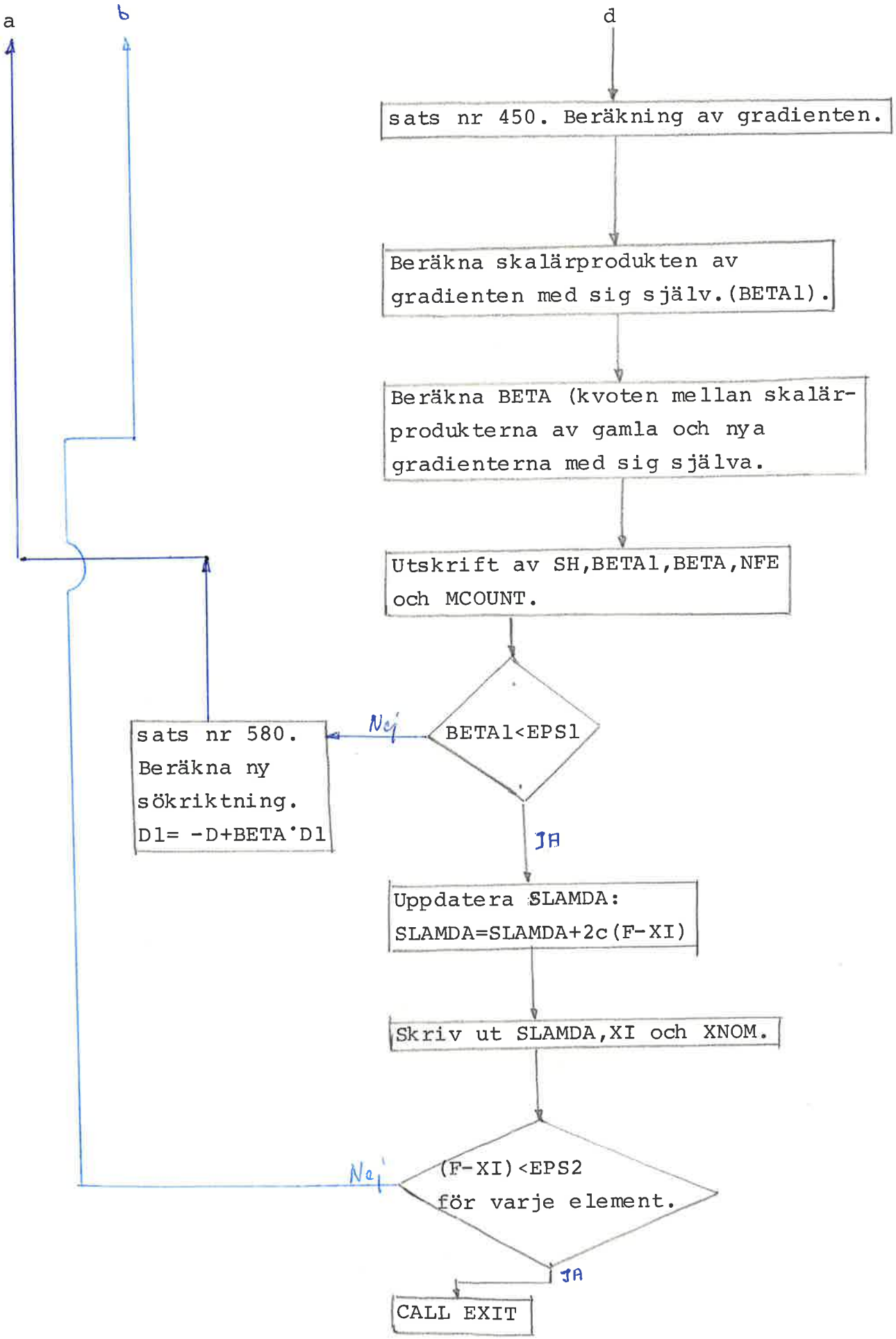
Flödesschemat baseras på den modifierade varianten av Fletcher-Reeves algoritm, som beskrivs i kapitel 4 och vars flödesschema finns i appendix A.
Flödesschemat visar hur programmet fungerar i stort utan att gå in på några beräkningar, dessa beskrivs i avsnitt 7,8,9,10.
Utskrifter och hopp är betonade, varför satsnummer från programmet har satts ut i flödesschemat vid varje hopp.
För förklarande av symboler mm och förtydligande av programmet se avsnitt 6,

Flödesschema för program IMF1.




a
b
c
Beräkning av SH samt utskrift av detta.
Beräkning av minimeringsgradienten GB samt utskrift av denna.
GB≤0
Nej
sats 235
GB= -GB
D1= -D1
JA
sats 240. MCOUNT nollställes.
sats nr 250. Start av extrapolation. MCOUNT=MCOUNT+1.
Beräkna nytt XI samt beräkna XNOM ur detta.
Beräkna nytt funktionsvärde SH samt GB. Skriv ut dessa.
a
b
d
c


a
b
d
c
Nej
MCOUNT>4
E
Interpolation eller extrapolation?
Ja
sats nr 316.
CALL EXIT
Felutskrift.
I
Utskrift av MCOUNT.
Nollställ NCOUNT (räknare av antalet interpolationer).
sats nr 230. Start av interpolation
NCOUNT=NCOUNT+1
Kubisk interpolation. Nya XI,XNOM, och SH beräknas. Skriv ut SH.
Hopp till gradientberäkning.
(sats nr 450).
Interpolations punktens värde mindre än båda ändpunkternas.
JA
NEJ
Ja
Nej
NCOUNT<4
Nej
Högra ändvärdet mindre än det vänstra.
Ja
sats nr 400: Minimum betraktas som funnet. Hopp till gradientberäkning.
a
b
d


a
b
d
sats nr 450. Beräkning av gradienten.
Beräkna skalärprodukten av gradienten med sig själv.(BETA1).
Beräkna BETA (kvoten mellan skalärprodukterna av gamla och nya gradienterna med sig själva.
Utskrift av SH,BETA1,BETA,NFE och MCOUNT.
BETA1<EPS1
Nej
sats nr 580. Beräkna ny sökriktning. D1= -D+BETA*D1
JA
Uppdatera SLAMDA: SLAMDA=SLAMDA+2c(F-XI)
Skriv ut SLAMDA,XI och XNOM.
(F-XI)<EPS2 för varje element.
Nej
JA
CALL EXIT

6. Förtydligande och förklarande av programmet.

I programmet approximerad tidsfunktionerna (t. ex. x(t), u(t)) med värden i ett antal ekvidistanta tidpunkter. Tidsintervallet indelas därför i NSTEP intervall. Varje funktion kommer då att motsvaras av ett fält med ett index, där varje komponent är funktiosvärdet vid en viss tidpunkt, Om man har en vektor, där varje komponent är en funktion kommer vektorn att motsvaras av ett fält med två index, medan en matris ger ett fält med tre index. Sista index hos dessa fält anger alltid tidpunkten och betecknas med I.
I sätts lika med 1 vid starttidpunkten T0 och lika med nstp=nstep+1 vid sluttidpunkten T1. I anger den aktuella tidpunkten T genom sambandet T=T0+(I-1)H, där H är intervallängden, Nedan anges sambanden mellan olika vektorer och motsvarande fält:

λ	$\leftrightarrow$	SLAMDA(K,I)	K=1,2,...,N
x	$\leftrightarrow$	XNOM(K,I)	K=1,2,...,N
u	$\leftrightarrow$	UNOM(K,I)	K=1,2,...,N
$\dot{x}$	$\leftrightarrow$	XPNOM(K,I)	K=1,2,...,N
$\xi=\begin{bmatrix}\dot{x}\\u\end{bmatrix}$	$\leftrightarrow$	XI(K,I)	K=1,2,...,N
Sökriktning d	$\leftrightarrow$	D1(K,I)	K=1,2,...,N+NU
Gradienten g map x Gradienten g map u	$\leftrightarrow$	D(K,I)	K=1,2,...,N+NU

Vidare motsvarar:

n	N		
m	NU		
n+m	NXI		
$x(t_0)$	XIN(k)	K=1,2,...,N	
$F(x(t_1))$	SFX(K)	K=1,2,...,N	
f^k	F(K)	K=1,2,...,N	
f^k_x	FX(K,L)	K=1,2,...,N	L=1,2,...,N

f_u^k ↔ FU(K,L) K=1,2,...,N L=1,2,...,N

L ↔ SL

L_u ↔ SLU(K) K=1,2,...,NU

t_0 ↔ T0

t_1 ↔ T1

$\langle g_{k+1}, g_{k+1} \rangle$ ↔ BETA1(NSTP)

$\beta_k = \frac{\langle g_{k+1}, g_{k+1} \rangle}{\langle g_k, g_k \rangle}$ ↔ BETA

ε_1 ↔ EPS1

ε_2 ↔ EPS2

Varje gång XI ändras skall nytt XNOM beräknas enligt:

$$XNOM(K,I)=XIN(K)+\int_{T0}^{T1} XI(K,I)dt \qquad K=1,2,...,N$$

I programmet används en subrutin QSF som fungerar som följer:

Subrutinen utför integrationen av en ekvidistant tabulerad funktion med hjälp av Simpsons regel. Om dessa funktionsvärden y_i är givna i ekvidistanta punkter $x_i=a+(i-1)h$ beräknar QSF fältet z där $z_i=z(x_i)=\int_a^{x_i} y(x)dx$ i=1,2,...,n.

Om funktionsvärdena läggs i fältet y, intervallängden läggs i h och antalte punkter läggs i NDIM fås fältet z genom anropet

```
CALL QSF(H,Y,Z,NDIM)
```

För närmare studium av subrutinen se (5).

Varje gång gradienten skall beräknas någonstans i programmet utförs detta genom att hoppa till gradientberäkningen. Återhopp sker genom väljarstyrt hopp.

Beskrivning av programmets funktion i stora drag:

Programmet startar med en gissning av XPNOM,UNOM, och SLAMDA. Dessa erhålles från subrutinen USER. En sökriktning beräknas och minimeringen längs denna göres.
Därefter testas om BETA1 dvs skalärprodukten av gradienten med sig själv är tillräckligt liten.
Om den är det uppdateras SLAMDA genom att skillnaden XPNOM-F multipliceras med 2c och läggs till det gamla SLAMDA-värdet varpå varje värde i matrisen XPNOM-F testas mot ett tal EPS2. Om något värde överstiger EPS2 omstartas programmet med negativa gradienten som sökriktning, i annat fall är körningen klar. Om BETA1 är för stor beräknas en ny sökriktnig och en ny minimering lägs denna göres etc.

Utskrifter:

Utskrifterna startar med
RESULTS AFTER MINIMIZING A FUNKTION J=SF(X(TF))+
INT(SL(X,U) UNDER THE CONDITION DX/DT=F(X,U)
WHERE X=(X1,X2,...,XN) AND U=U(U1,U2,...,UM)
TF IS THE TIME OF THE FINAL STATE
INITIAL VALUES
Därefter följer utskrifter på värden av
N NU NSTEP H T0 T1 NPRINT C EPS1 EPS2
Sedan skrivs UNOM XPNOM SLAMDA och XNOM ut.
Kolumnen längst till vänster är tiden T.
BETA1= skalärprodukten av gradienten med sig själv.
STEP= steglängden vid extrapolation.

Nu börjar minimeringen.
COMPUTED VALUES DURING MINIMISATION
SH= första funktionsvärdet
FIRST VALUE OF GB:
GB= minimeringsgradienten i första punkten
SH COMPUTED AFTER EXTRAPOLATION: SH=
GB COMPUTED AFTER EXTRAPOLATION NEW XI: GB=
Detta är funktionsvärdet och minimeringsgradienten i den framextrapolerade punkten.
Extrapolationen är färdig när
MCOUNT= skrivs ut. MCOUNT anger hur många extrapolationer som gjorts.
Interpolationen startar me utskriften
START INTERPOLATION
STEPLENGTH WHILE INTERPOLATING: STG=
Interpolationen går baklänges och STG är steglängden.
Därefter skrivs det framinterpolerade funktionsvärdet ut
SH COMPUTED WHILE INTERPOLATING: SH=
GB=
Sedan testas om minimum är funnet och man kan snabbt se om så är fallet genom utskriften
MINIMUM FOUND? SH= HA= HB=
Om minimum hittats blir nästa utskrift
VALUES AFTER MINIMIZING
NFE=
NCOUNT=
STEP=
BETA1=
BETA=

Om nu BETA1<EPS1 blir nästa utskrift XI, XNØM och nya SLAMDA annars börjar det om igen med COMPUTED VALUES DURING MINIMISATION.
Efter utskriften av SLAMDA är antingen programmet slut eller också börjar en ny minimering.
Det kan även vara lämpligt att skriva ut XI och XNOM efter varje endimensionell minimering om inte SLAMDA uppdateras ofta för då kan körtiden ta slut utan att några styrstrategier blir utskrivna.

7. Minimering längs en sökriktning.

Flödesschema se appendix B.
Förklaring av symboler i flödesschemat:

NFE= antalet funktionsevalueringar.
STEP= steglängden vid extrapolation.
ITA= interpolationsintervallets längd.
MCOUNT= antalet extrapolationer under en minimering längs en sökriktning.
NCOUNT= motsvarande antal interpolationer.
SH= aktuellt funktionsvärde.
HB= funktionsvärdet i högra ändpunkten av intervallet,
HA= funktionsvärdet i vänstra ändpunkten av intervallet.

XI= $\begin{bmatrix} \dot{x} \\ u \end{bmatrix}$

GB= minimeringsgradienten = skalärprodukten av gradienten och sökriktningen.
D1= sökriktning

Funktion:
Vi minimerar längs en riktning D1 genom att ändra XI enligt XI=XI+ALFA*D1.
Metoden börjar med att beräkna funktionsvärde och minimeringsgradient för givet XI. Om minimeringsgradienten är större än noll sättes GB= -GB och D1= -D1. Anledningen till detta är att metoden endast kan extrapolera åt "höger" och om GB>0 vill vi extrapolera åt vänster för att nå minimum. Se figur 1.
Därefter göres en extrapolation med steglängden STEP. Funktionsvärde och minimeringsgradienten beräknas i den nya punkten B (se figur 2). Om HA<HB eller GB>0 (GB är minimeringsgradienten i punkten B) så startas interpolationen.

I annat fall multipliceras steglängden med fyra (se avsnittet förbättringar) och en ny extrapolation göres. Om 20 extrapolationer gjorts i rad avbrytes programmet och Mcount .GT.20 skrives ut.
Innan interpolationen börjar skrivs antalet extrapolationer ut (NCOUNT=).
Därefter göres en kubisk interpolation varvid man anpassar ett tredjegradspolynom till kurvan.
Därvid beräknas storheterna Z,W samt STG, där STG anger hur lågt steg tillbaks man skall ta vid interpolationen (se figur 3). I den nya punkten C (se figur 4) beräknas funktionsvärdet. Om detta är mindre än både HA och HB tages denna punkt som minimum, om inte, divideras step med fyra och om HB<HA väljes HB till minimumpunkt , om inte, interpoleras ytterligare en gång, varvid interpolationen utföres mellan punkterna A och C.
Vitsen med att STEP multipliceras respektive divideras med fyra är att steglängden ändras så att vid nästa minimering färre extrapolationer och interpolationer behöver göras.
Om mer än fyra interpolationer göres i rad avbrytes minimeringen och omstart av Fletcher-Reeves göres med sökriktningen lika med negativa gradienten.

Kommentar:

Avsikten med minimeringen längs en riktning är att finna en punkt i närheten av ett minimum. Detta behöver inte vara det globala minimat.
För att minimeringen skall fungera tillfredställande får funktionen inte vara allt för komplicerad.
Vid genomförandet av den kubiska interpolationen kan uttrycket $Z^2-GA\cdot GB$, vilket står under ett rottecken bli negativt. I så fall sättes detta till noll.
Efter minimeringen skrivs följande ut:

```
VALUES AFTER MINIMIZING
NFE=  MCOUNT=  STEP=
```

BETA1=

BETA=

Merän fem omstarter av F-R pga misslyckade interpolationer tolereras inte. Utskriften blir:

TOO MANY RESTARTS MSTRT=

varefter körningen brytes.

8. Beskrivning av subrutin USER.

För att kunna använda programmet allmänt har vi en subrutin USER som innehåller alla data för ett specifikt problem. För att slippa anropa subrutinen med parametrar har den och huvudprogrammet en gemensam minnesarea, ett commonblock där värden på aktuella parametrar finns.
Subrutinen USER används dels för att ge initialvärden till huvudprogrammet dels för beräkningar av exempelvis funktionsvärden och partiella derivator. Dessa beräkningar gäller alltid för en bestämd tidpunkt varför XNOM och UNOM inte används där. Istället läggs XN/M och UNOM för den aktuella tidpunkten över i XVEC och UVEC vilka har gemensam minnesarea med X och U i subrutinen, varefter beräkningarne kan göras.
Subrutinen USER består av ett antal underavdelningar med olika entry points, vilka kan anropas var för sig med en CALL-instruktion.

Beskrivning av de olika avdelningarnas användning:

1. INIT
Här finns alla initialvärden. Se appendix C.

N	anger antalet tillståndsfunktioner.
NU	anger antalet styrfunktioner.
NSTEP	anger antalet intervall som det aktuella tidsintervallet från T0 till T1 indelats i.
T0	anger starttidpunkt.
T1	anger sluttidpunkten.
H	är intervallängden=(T1-T0)/NSTEP. (OBS Glöm ej ändra H när du ändrar NSTEP)
NPRINT	anger hur många punkter man vill ha utskrivna. Om t. ex. NPRINT sättes till 10 skrivs var tionde punkt ut.
XIN	är randvärdena för XNOM($X(t_0)$)

Begynnelsevärdena på XPNOM,SLAMDA och UNOM placeras också här.
EPS1 anger hur liten skalärprodukten av gradienten med sig själv måste vara för att minimeringen skall vara färdig för givet SLAMDA.
EPS2 anger hur litet varje värde i matrisen XI-F måste vara innan programmet betraktas som färdigt och körningen avbryts.
C anger hur hårt avvikelse från ekvationen XI=F (x≒f) skall straffas i SH (funktionen).
STEP är steglängden vid extrapolation första gången vi minimerar längs en riktning i programmet.

2. FINLOS

SF är den funktion i SH som endast beror av X:s sluttillstånd (F).

3. INTLOS

SL är den funktion i SH som står under integralstecknet.

4. BBOUND:

SFX är partiella derivatan av SF med avseende på x.

5. PDER:

Här står partiella derivatorna av F och SL map x och u (FX,FU,SLX och SLU).

6. SYSTEM:

Här anges systemekvationerna F (f).

9. Gradientberäkning.

Gradienten mapxi består av två delar:

a) Gradienten map u.

Teorin ger:

$$\nabla_u H = L_u^T + f_u^T \lambda + 2c f_u^T (f - \dot{x})$$

$\nabla_u H$ är en kolonnvektor med m (NU) komponenter.
L_u är en radvektor med m komponenter.
f_u är en nxm (NxNU) matris.
λ,f och x är kolonnvektorer med n (N) komponenter.
Observera att varje komponent är en funktion.
För en viss tidpunkt gäller:

$$\nabla_u H = \begin{bmatrix} L_{ul} + \sum_{i=1}^{n} f_{ul}^i \; \lambda_i + \sum_{i=1}^{n} f_{ul}^i \; (f^i - \dot{x}^i) \\ \cdot \\ \cdot \\ L_{um} + \sum_{i=1}^{n} f_{um}^i \; \lambda^i + \sum_{i=1}^{n} f_{um}^i \; (f^i - \dot{x}^i) \end{bmatrix}$$

Med utgångspunkt från denna matris har nedanstående programavsnitt skrivits:

```
C       GRAD WITH RESPECT TO UNOM
C
        DO 500 I=1,NSTP
        T=T0+(I-1)*H
        DO 485 JU=1,NU
        JK=JU+N
485     UVEC(JU)=XI(JK,I)
        DO 490 J=1,N
490     XVEC(J)=XNOM(J,I)
        CALL PDER
        CALL SYSTEM
        DO 500 KU=1,NU
        KN=KU+N
        D(KN,I)=0.
        DO 495 K=1,N
495     D(KN,I)=D(KN,I)+FU(K,KU)*SLAMDA(K,I)+2.*C*FU(K,KU)*(F(K)-XI(K,I))
500     D(KN,I)=D(KN,I)+SLU(KU)
```

b) Gradienten map $\dot{x}$.

Teorin ger:

$$\nabla_{\dot{x}}H=F_x^T(x(t_1))-\lambda-2c(f-\dot{x})+\int_t^{t_1}(L_x^T+f_x^T\lambda+2cf_x^T(f-\dot{x}))ds$$

F_x och L_x är radvektorer med n (N) komponenter.
f_x är en nxm (NxM) matris.
För en viss tidpunkt gäller:

$$\nabla_{\dot{x}}H=\begin{bmatrix} F_{x1}(x(t_1))-2c(f^1-\dot{x}^1)-\lambda^1+\int_t^{t_1}(L_{x1}+2c\sum_{i=1}^{n}(f^i_{x1}(f^i-\dot{x}^i)+f^i_{x1}\lambda^i))ds \\ \cdot \\ \cdot \\ F_{xn}(x(t_1))-2c(f^n-\dot{x}^n)-\lambda^n+\int_t^{t_1}(L_{xn}+2c\sum_{i=1}^{n}(f^i_{xn}(f^i-\dot{x}^i)+f^i_{xn}\lambda^i))ds \end{bmatrix}$$

Med utgångspunkt från denna matris har nedanstående program-avsnitt skrivits:

```
C       GRAD WITH RESPECT TO XPNOM
450     T=T1
        I=NSTP
        DO 430 J=1,N
430     XVEC(J)=XNOM(J,I)
        DO 435 JU=1,NU
        JK=JU+N
435     UVEC(JU)=XI(JK,I)
        CALL BBOUND
```

SFX får sina värden genom anrop av subrutin USER (entry point BBOUND) vi sluttidpunkten (I=NSTP).

```
      DO 480 KP=1,N
      DO 466 I=1,NSTP
      T=T0+(I-1)*H
      DO 455 J=1,N
455   XVEC(J)=XNOM(J,I)
      DO 460 JU=1,NU
      JK=JU+N
460   UVEC(JU)=XI(JK,I)
      CALL SYSTEM
      CALL PDER
      IK=NSTP-I+1
      ETA(IK)=0
      DO 465 K=1,N
C     BACKWORDS INTEGRATION
445   ETA(IK)=ETA(IK)+FX(K,KP)*SLAMDA(K,I)+2.*C*FY(K,KP)*(F(K)-XI(K,I))
446   ETA(IK)=ETA(IK)+SLX(KP)
      CALL QSF(H,ETA,SINT,NSTP)
```

För varje komponent av gradienten beräknas uttrycket under integraltecknet och läggs i vektorn ETA. Eftersom QSF endast beräknar integraler från t_0 till t där t varierar, och vi vill beräkna integralen från t till t_1, måste vektorn ETA vara vänd. QSF anropas varvid ETA integreras och resultatet läggs i SINT. Slutligen måste SINT vändas för att ge rätt resultat i slutberäkningarna.

```
      DO 480 I=1,NSTP
      T=T0+(I-1)*H
      DO 470 J=1,N
470   XVEC(J)=XNOM(J,I)
      DO 475 JU=1,NU
      JK=JU+N
475   UVEC(JU)=XI(JK,I)
      CALL SYSTEM
      IK=NSTP-I+1
480   D(KP,I)=SINT(IK)-SLAMDA(KP,I)-2.*C*(F(KP)-XI(KP,I))+SFX(KP)
```

10. Beskrivning av beräkningen av SH.

Funktionen H kallas i programmet för SH för att undvika förväxling med steglängden H.

$$SH=F(x(t_1))+\int_{t_0}^{t_1} L(x,u)+\lambda^T(f-\dot{x})+c(f-\dot{x})^T(f-\dot{x})ds$$

Beräkningen av SH göres i nedanstående programavsnitt.

```
C       COMPUTE NEW SH
C
        DO 370 I=1,NSTP
        T=T0+(I-1)*H
        DO 355 J=1,N
355     XVEC(J)=XNOM(J,I)
        DO 360 JU=1,NU
        JK=JU+N
360     UVEC(JU)=XI(JK,I)
        CALL SYSTEM
        CALL INTLOS
        P=0.
        DO 365 K=1,N
365     P=SLAMDA(K,I)*(F(K)-XI(K,I))+P+C*(XI(K,I)-F(K))**2
370     E(I)=P+SL
         CALL QSF(H,E,SINT,NSTP)
        CALL FINLOS
        SH=SINT(NSTP)+SF
```

Först sker en överlagring av information i arbetsvektorer. Därefter beräknas det som står under integraltecknet för alla tidpunkter och läggs i vektorn E varefter QSF anropas dvs E integreras och resultatet läggs i SINT(NSTP). Därefter adderas SF (F) till SINT.

11. Testexempel.

Testexempel 1.

System:
$$\frac{dx_1}{dt} = x_2 \qquad x_1(0)=2$$

$$\frac{dx_2}{dt} = u_1 \qquad x_2(0)=2$$

Förlustfunktion:
$$SH=12\cdot x_1(2)-10\cdot x_2(2)+\int_0^2 u_1^2 dt$$

Optimal lösning:
$$u_1^*(t)=6t-7$$
$$x_1^*(t)=t^3-7/2\cdot t^2+2t+2$$
$$x_2^*(t)=3t^2-7t+2$$

$$SH_{min}=26$$

Med c=1
erhölls värdet SH=25.83
efter 12 uppdateringar av SLAMDA och 23 iterationer där iteration= minimering längs en sökriktning.
Värdet på BETA1 var då 0.001.
NSTEP=500 användes för att få så god nogrannhet som möjligt.
Startvärdet på EPS1 var 10.

Testexempel 2.

System:
$$\frac{dx_1}{dt} = -x_1+u_1+u_2 \qquad x_1(0)=1$$

$$\frac{dx_2}{dt} = x_1-2x_2+u_1 \qquad x_2(0)=1$$

Förlustfunktion:
$$SH=x_1^2(1)+x_2^2(1)+\int_0^1 (4x_1^2+5x_2^2+u_1^2+u_2^2)ds$$

Optimal lösning:
$$u_1^*(t)= -e^{-3t}\ (2-t)$$

$$u_2^*(t)= -e^{-3t}\ (1-t)$$

$$x_1^*(t)= e^{-3t}\ (1-t)$$

$$x_2^*(t)= e^{-3t}$$

$$SH_{min}=2$$

För följande parametervärden erhölls SH=1.982
EPS1=0.01
NSTEP=500
c=10
efter 30 iterationer och 10 uppdateringar av SLAMDA.
Vid minimum var värdet på BETA1=0.0011
Men redan efter 15 iterationer och en uppdatering av
SLAMDA var värdet på SH=1.968.

Testexempel 3.

System: $\frac{dx_1}{dt} = u_1$ $\qquad x_1(0) = -1$

$\frac{dx_2}{dt} = x_1^2 + 2x_1 \cdot e^t$ $\qquad x_2(0) = 0$

Förlustfunktion: $SH = x_2(1) + \int_0^1 u_1^2 dt$

Optimal lösning: $u_1^*(t) = \frac{1}{2} e^t (t-1)$

$x_1^*(t) = e^t (\frac{t}{2} - 1)$

$x_2^*(t) = e^{2t} (\frac{t^2}{8} - \frac{t}{8} - \frac{7}{16}) + \frac{7}{16}$

$SH_{min} = -2.6459$

Vid körning på dator erhölls följande resultat då
EPS1=0.1
NSTEP=500
c=8
efter fem uppdateringar av SLAMDA och 26 iterationer
SH= -2.638 .
BETA1 var då 0.021

12. Jämförelseexempel.

System:

$$\frac{dx_1}{dt} = (1-x_2^2)\,x_1 - x_2 + u \qquad x_1(0)=0$$

$$\frac{dx_2}{dt} = x_1 \qquad x_2(0)=1$$

Förlustfunktion:

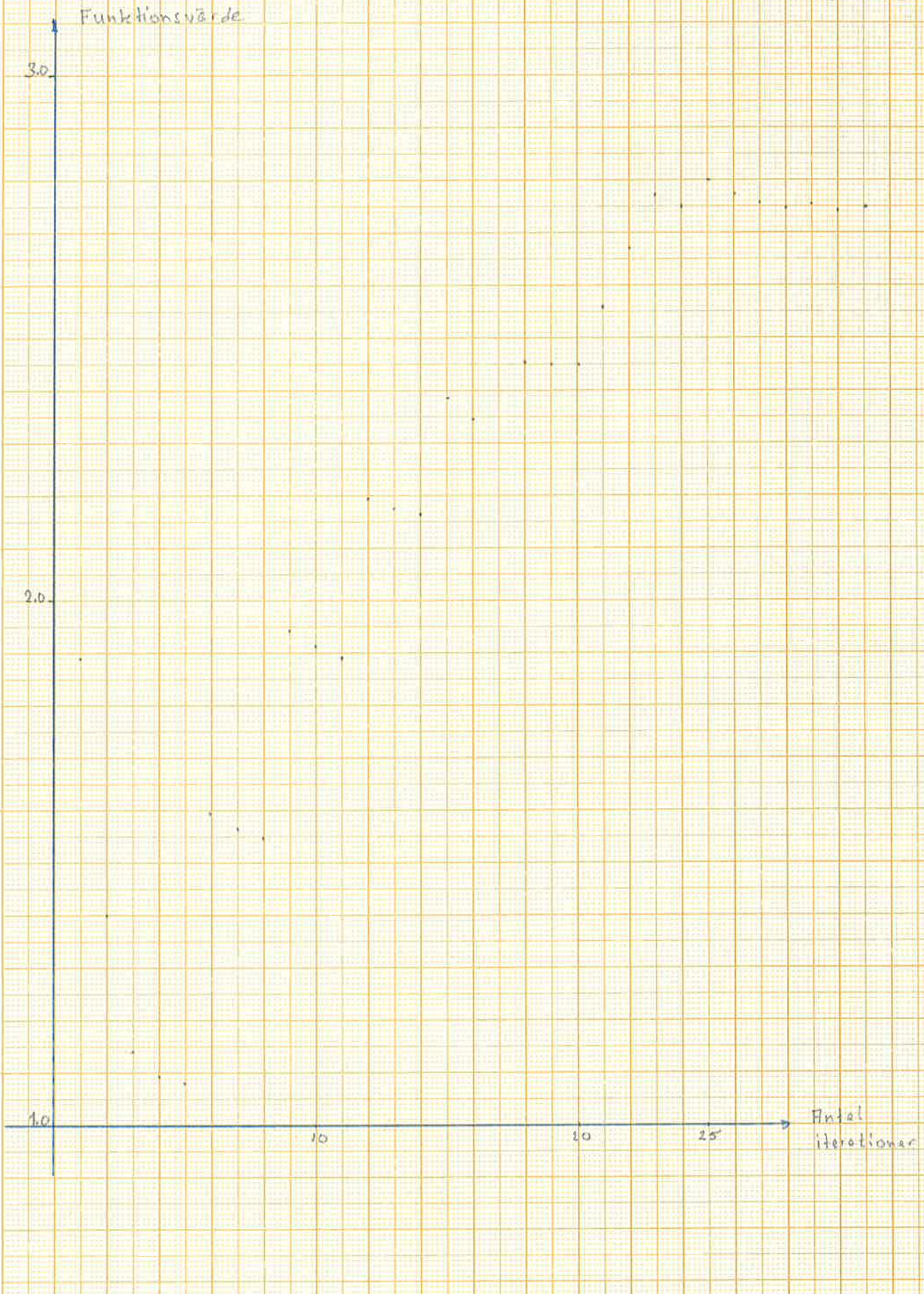
$$SH = \int_0^5 (x_1^2+x_2^2+u^2)\,dt$$

Som exempel på hur subrutinen USER används ges i appendix C den USER, som använts för detta exempel.
För att kunna jämföra metoden med andra, har i diagram 1 avsatts SH som funktion av antalet iterationer för olika SLAMDA-värden.
I diagram 1 ser man hur funktionsvärdena först sjunker med antalet iterationer och sedan gör ett tvärt hopp uppåt när SLAMDA-värdena uppdateras varefter programmet söker minimum för dessa SLAMDA-värden etc.
Man kommer alltså att nalkas minimum underifrån.

Referenserna (1) och (2) angriper samma problem med andra metoder.

Diagram 1.
Jämförelseexempel.

SH som funktion av antalet iteratio ner.



13. Tillämpningsexempel.

Problemet härstammar från en containerterminal. När ett skepp lastar av, förs containern först med en kajkran till en väntande lastbil, vilken kör till ett lager, där en annan kran sätter containern på en förutbestämd plats. Sedan kör den tomma lastbilen tillbaka till kajkranen.

Genom att minimera tiderna för överföring med kranarna kan lossningstiden för båten minskas, vilket är ekonomiskt fördelaktigt.
Då de två kranarna är ganska lika behandlas endast lagerkranen här.

I den modell som använts, antas styrvariablerna vara accelerationen hos vagnen och vinschen. Detta motsvarar en kran gjord för manuell styrning, varvid kranen är utrustad med olika regulatorer för att göra styrningen oberoende av lastens massa.

Den överföring med kran som minimerats visas i figur 5. När lastbilen anländer antas traversen vara belägen så att den slutliga överföringen kan göras av vagn och vinsch. Problemet reduceras då till två dimensioner.

Det antas, att avståndet mellan last och vagn ursprungligen är 12 meter, att containern kan anses som en punktmassa och att tyngdpunkten ligger i det geometriska centret. Alla avstånd refererar till denna punkt.

När överföringen börjar, befinner sig vagnen rakt över lastbilen och står stilla. Den vertikala hastigheten hos lasten är också noll. Vid slutet av överföringen antas på samma sätt systemet vara i vila.

Genom att sätta upp en modell för systemet och införa tillstånds- och styrvariabler enligt nedan erhålls följande system:

$$\dot{x}_1 = x_2 \qquad x_1(0)=0$$

$$\dot{x}_2 = u_1 \qquad x_2(0)=0$$

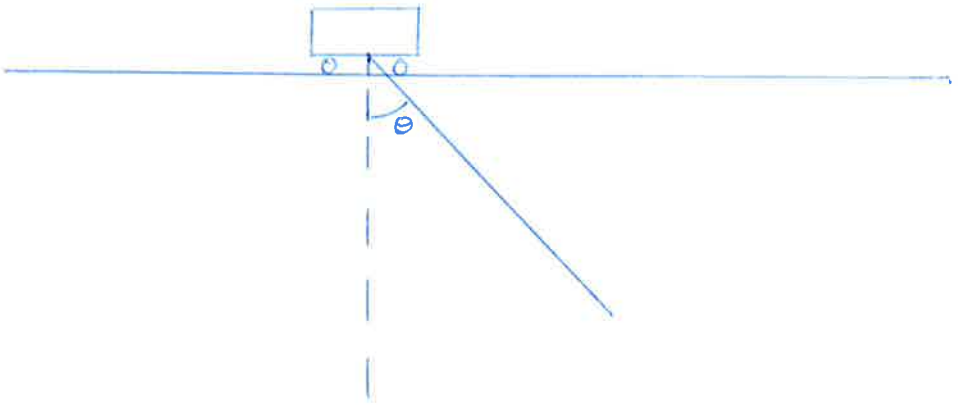
$$\dot{x}_3 = x_4 \qquad x_3(0)=0$$

$$\dot{x}_4 = -g\frac{\sin x_3}{x_5} - \frac{2\,x_4\,x_6}{x_5} - \frac{u_1 \cos x_3}{x_5} \qquad x_4(0)=0$$

$$\dot{x}_5 = x_6 \qquad x_5(0)=12$$

$$\dot{x}_6 = u_2 \qquad x_6(0)=0$$

där x_1= trallans läge x_2= trallans hastighet $x_3=\theta$, $x_4=\dot{\theta}$, x_5= linans längd, x_6= vinschhastigheten, u_1= trallans acceleration, u_2= vinschens acceleration.



Under förutsättningarna att t_1 är känd (t_1=20 s), sluttillståndet enligt figur 5 och att vi vill minimera den energi som åtgår vid förflyttningen, erhålles följande förlustfunktion:

$$SH=10((x_1-12)^2+x_2^2+x_3^2+x_4^2+(x_5-6)^2+x_6^2)+\int_0^{20}(u_1^2+u_2^2)dt$$

Förutsättningen att containern skall vara i vila i sluttillståndet tas om hand av första delen av SH, där avvikelse straffas.
Då $u_1^2+u_2^2$ är proportionellt mot den energi som åtgår, har denna funktion valts som L.

Vid körning på dator erhölls de styrstrategier som visas i diagram 2.
NSTEP valdes till 100 för att få rimliga körtider.

Diagram 2.
Tillämpnings-
exempel.

De optimala x_1 och x_2 som funktion av tiden.

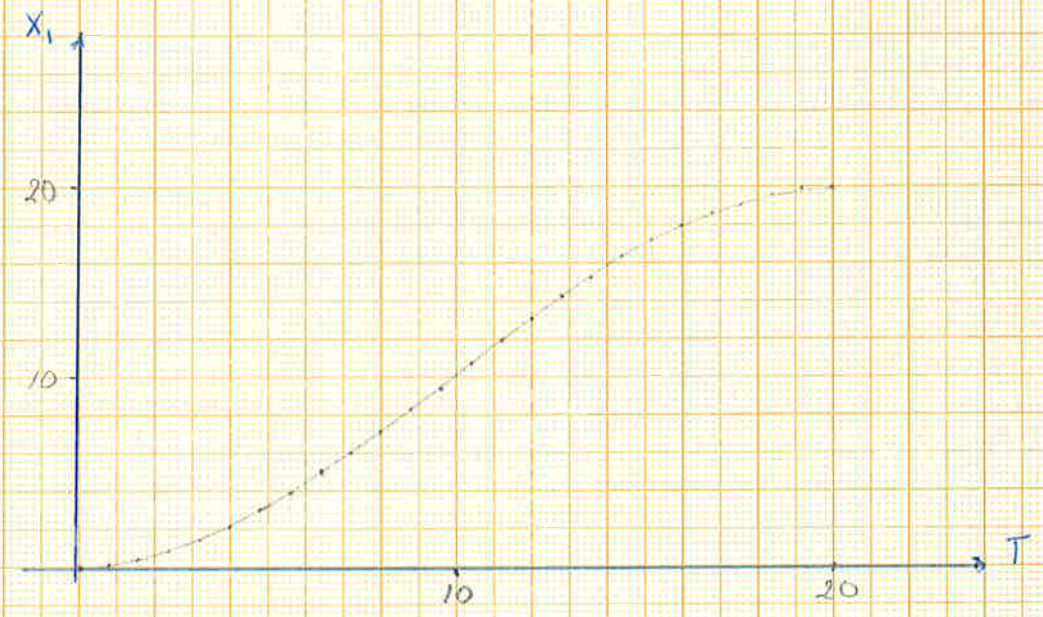
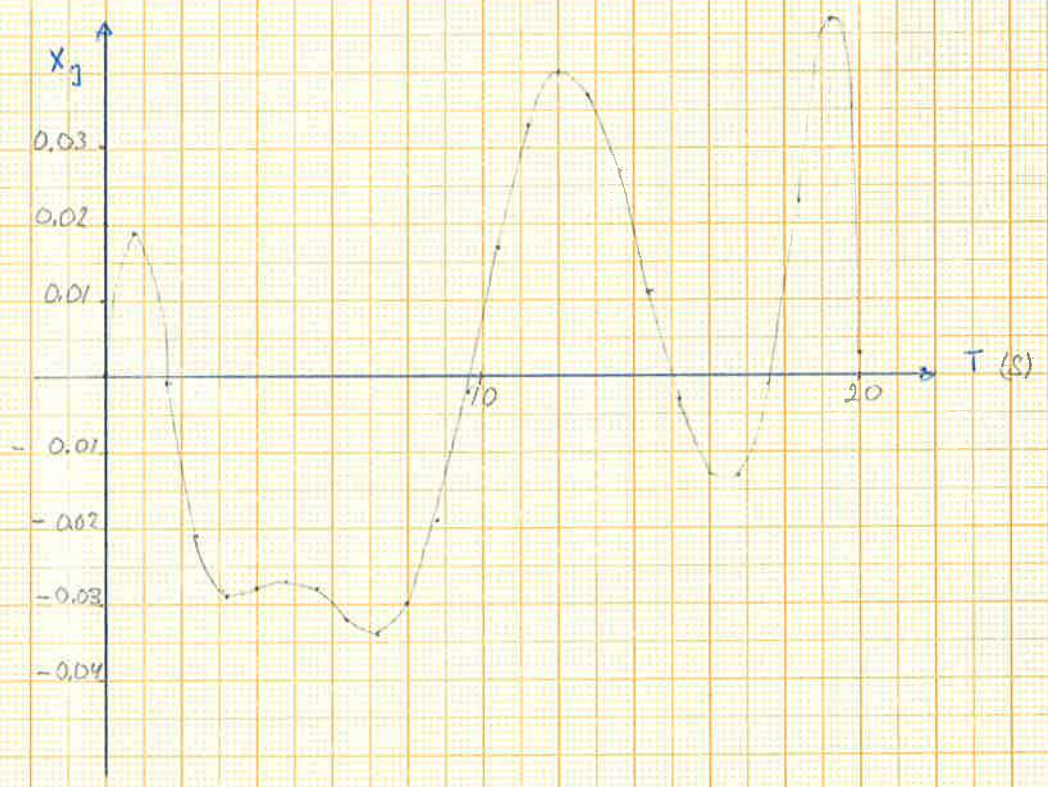


Diagram 3.
Tillämpnings-
exempel.

De optimala x_3 och x_4 som funktion av tiden.



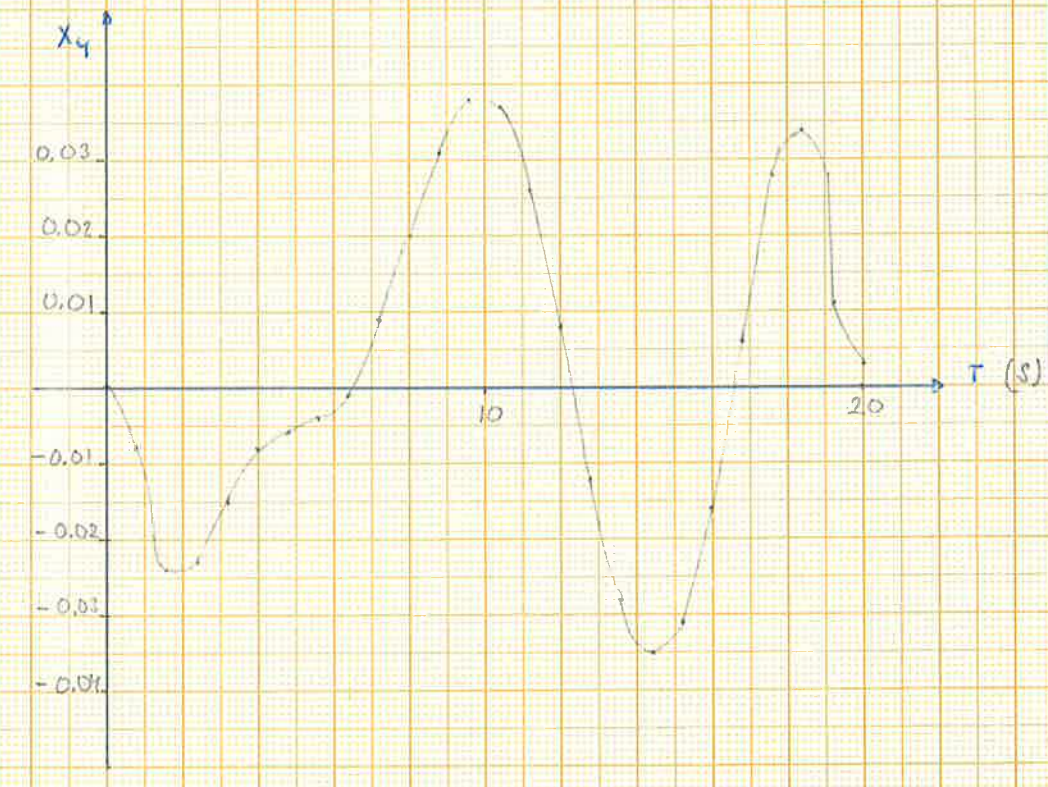
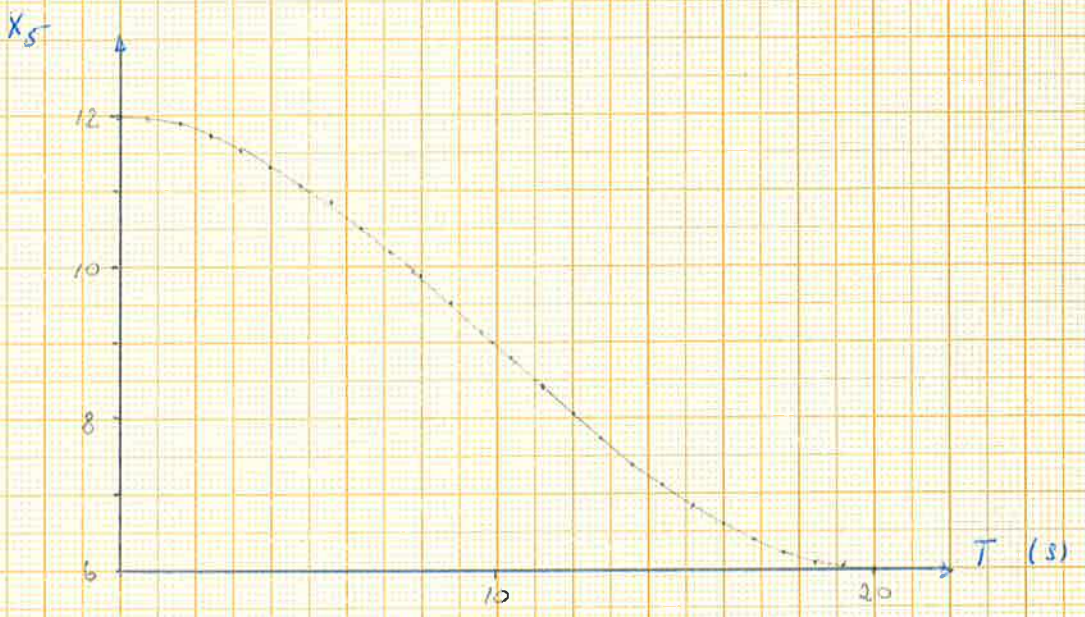


Diagram 4.
Tillämpnings-
exempel.

De optimala x_5 och x_6 som funktion av tiden.



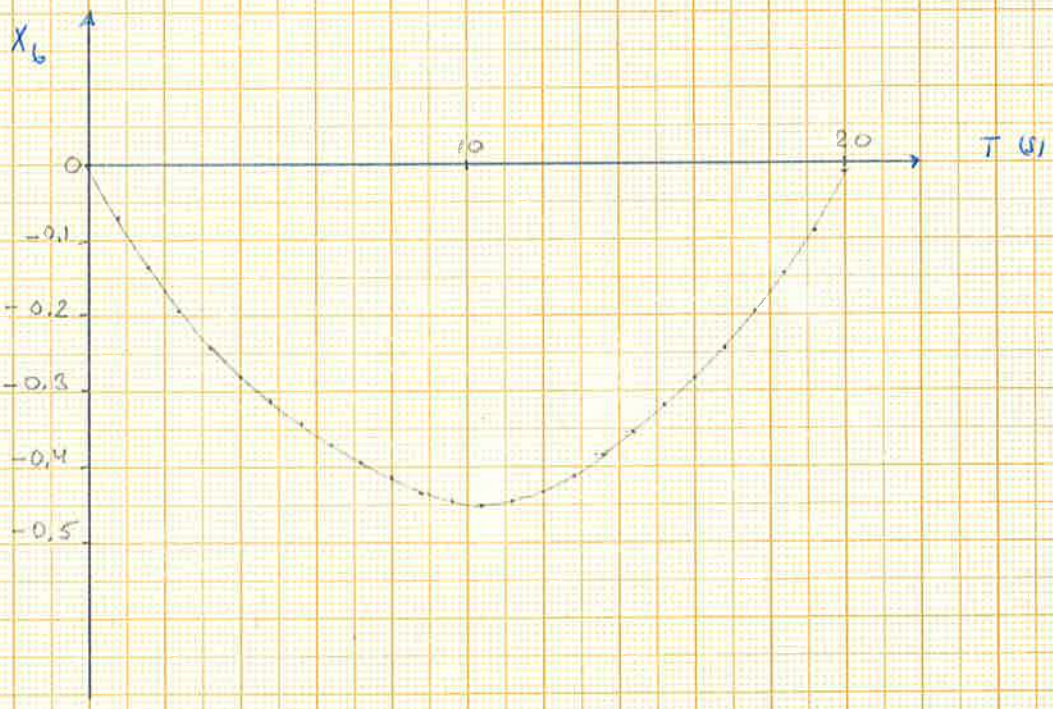
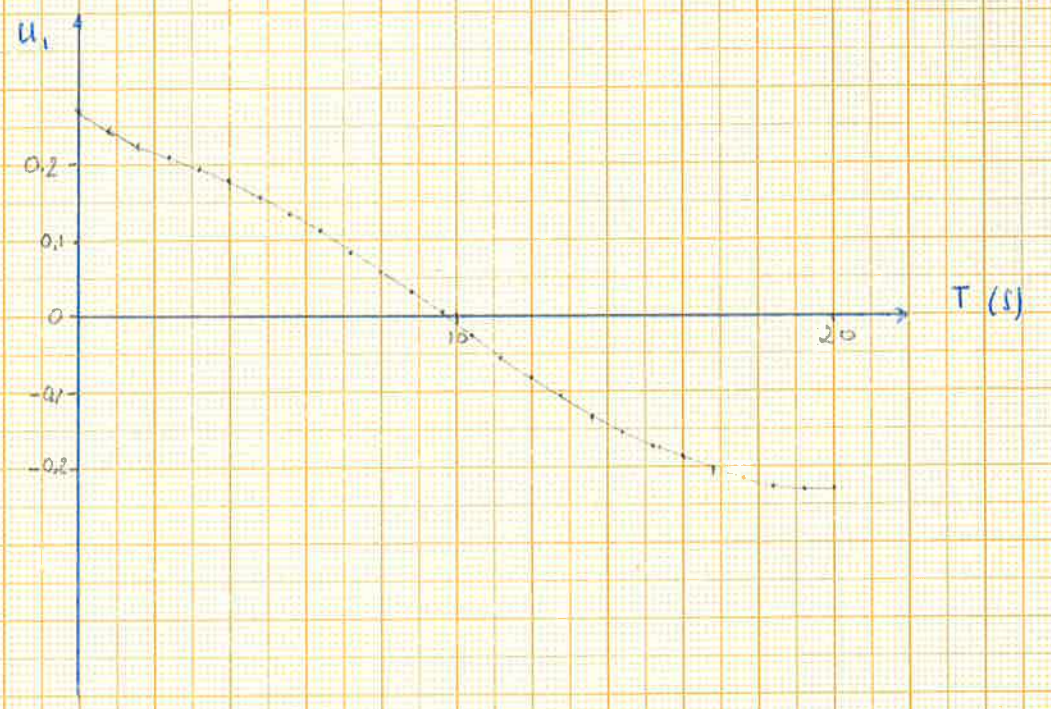
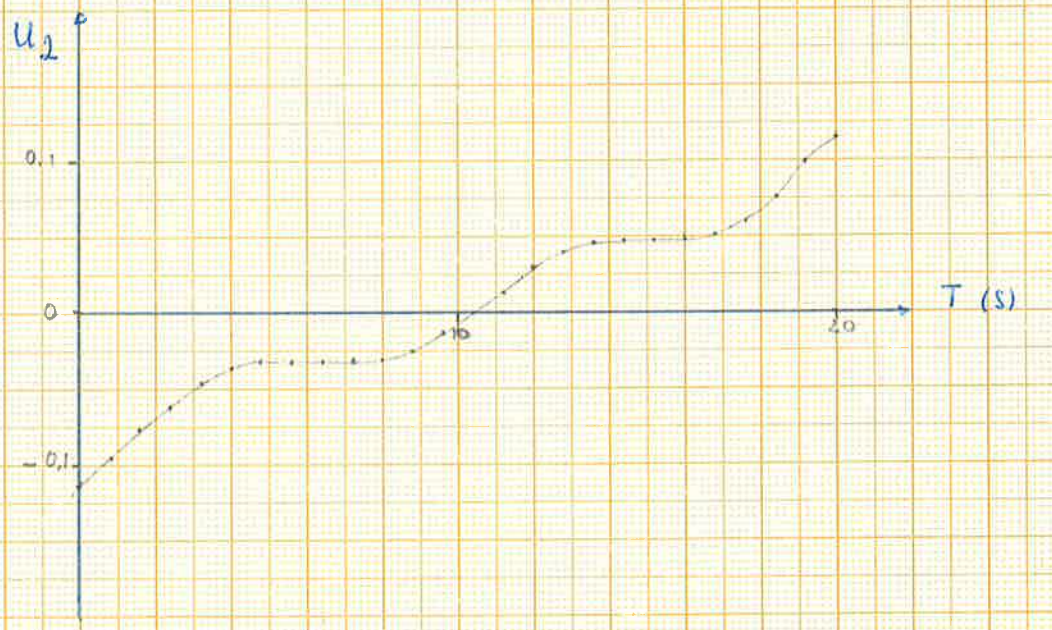


Diagram 5.
Tillämpnings-
exempel.

De optimala u_1 och u_2 som funktion av tiden.





14, Egna erfarenheter och förbättringar i programmet.

Jag tyckte att det var för grovt att multiplicera steglängden med fyra varje gång extrapolationen misslyckades, eftersom man får onödigt stora intervall att interpolera på. Istället för att multiplicera med fyra använde jag faktorn $\frac{GA-GB}{4\ GA} + 4\ \frac{GB}{GA}$ vilken har fördelen att om man ligger nära minimum (GB=0) så blir andra termen i uttrycket mycket liten och den första ungefär en fjärdedel dvs steglängden blir 4 ggr kortare. Långt från minimum (GB=GA) multipliceras steglängden med 4. Interpolationsintervallet blir alltså mindre och det har visat sig att det sällan behövs mer än en interpolation för att nå minimum.

Man strävar ju efter att få så snabb konvergens som möjligt och därvid är det viktigt att välja parametrarna på rätt sätt. Om man använder ett stort c (t. ex 10) kommer konvergensen långt från minimum att bli dålig. Detta inses av den tredimensionella modellen av problemet. För stora c kommer minimum att ligga i en ränna med branta sidor.

Jag har i exempel 5 använt ett stort EPS1 (100) från början och sedan dividerat EPS1 med 3 efter varje uppdatering av SLAMDA vilket tycks ha snabbat upp minimeringen.

Var femte gång en ny sökriktning bildas för den endimensionella minimeringen, sätts riktningen lika med negativa gradienten. Detta har ökat konvergenshastigheten betydligt.

15. Jämförelse mellan olika metoder.

Vi vill först och främst påpeka att det är svårt att göra någon direkt jämförelse mellan metoderna (1),(2) och den beskriven här.
Detta beror på att metoderna arbetar på olika sätt.
Dessutom kan man ändra konvergenshastigheten genom att ändra på vissa parametrar (gäller ej (2).
Det är svårt att hitta värden på parametrarna så att optimal konvergenshastighet erhålles, dessutom gäller dessa värden endast för ett specifikt problem.
Vi tar upp metod för metod och diskuterar dessas för och nackdelar och jämför med andra metoder:

Metod 1 (se referens 1):
Se diagram i referens 1.
Samma typ av konvergens erhålles som med min metod.
Konvergenshastigheten verkar vara ungefär densamma som i min metod men väsentligt sämre än i metod (2).
En nackdel är att det är svårt att hitta de bästa värdena på EPS 1.
Fördel: Problemet kan lösas utan att systemekvationen löses och metoden kan enkelt generaliseras till mera komplexa problem, t. ex. system med tidsfördröjningar.

Min metod:

Se diagram 1.

Fördel: Systemekvationen behöver ej lösas och metoden kan lätt generaliseras till mera komplexa problem t. ex. system med tidsfördröjningar.

Nackdelar: Svårigheten att finna sådana värden på EPS1 så att onödigt räknearbete undvikes.

Dessutom måste ett $c>c_0$ hittas, så att minimum kan erhållas.

Detta c får ej heller vara för stort för då blir konvergenshastigheten långt från minimum låg.

Metod 2 (se referens 2):

Se diagram 1, referens (1).

Nackdelar: Systemekvationen måste gå att lösa.

Fördel: Bra konvergenshastighet på samtliga exempel.

Eftersom iterationerna endast sker på u blir minimeringen enklare.

Dessa jämförelser gäller det jämförelseexempel som behandlatd i avsnitt 12, men stämmer bra även på övriga exempel.

Överföringen av metoderna på program för dator, har tagit ungefär lika lång tid. Programmen har blivit ungefär lika långa och eftersom samma huvudprinciper utnyttjas har programmen till stora delar blivit identiska.

16. Referenser.

1 Persson R, "Straff-funktionsmetoder för numerisk lösning av optimala styrproblem". Examensarbete på Institutionen för Reglerteknik, Lunds Tekniska Högskola.

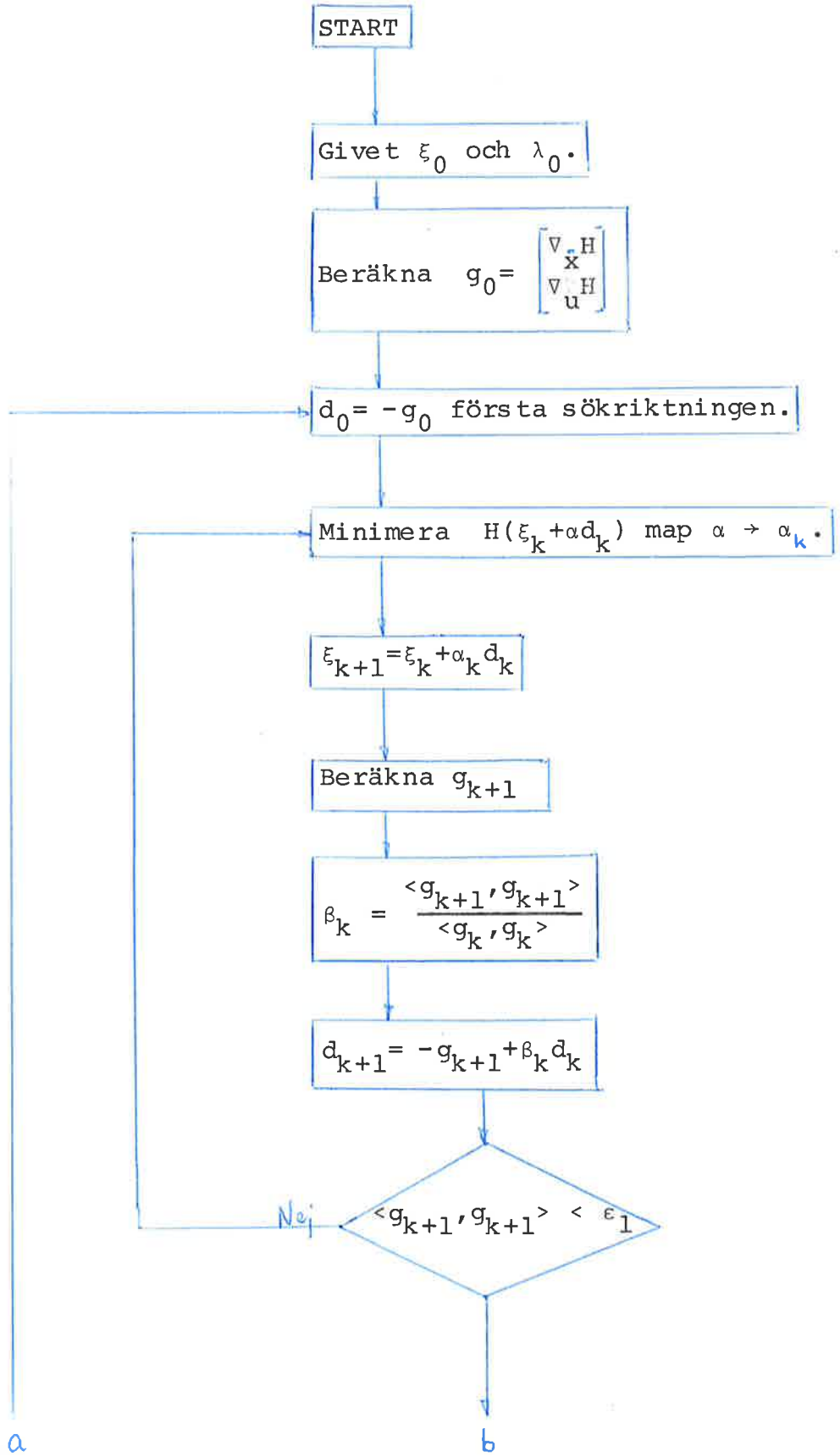
2 Persson L, "Konjugerade gradientmetoden för optimala styrproblem". Examensarbete på Institutionen för Reglerteknik, Lunds Tekniska Högskola.

3 Mårtensson K, "New approach to constrained function optimization", rapport 7206, Institutionen för Reglerteknik Lunds Tekniska Högskola.

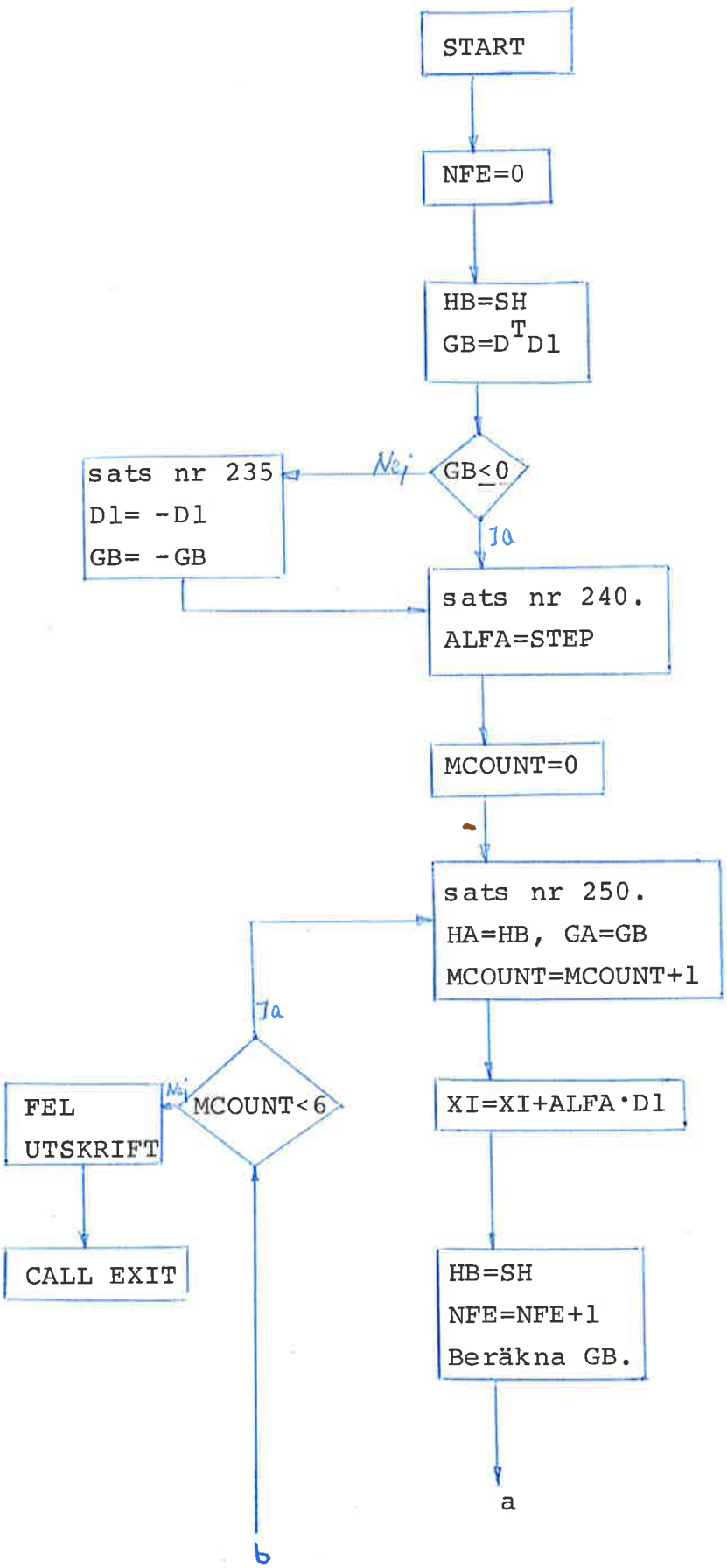
4 Luenberger D.G., "Introduction to linear and nonlinear programming", Addisson Wesley,1973.

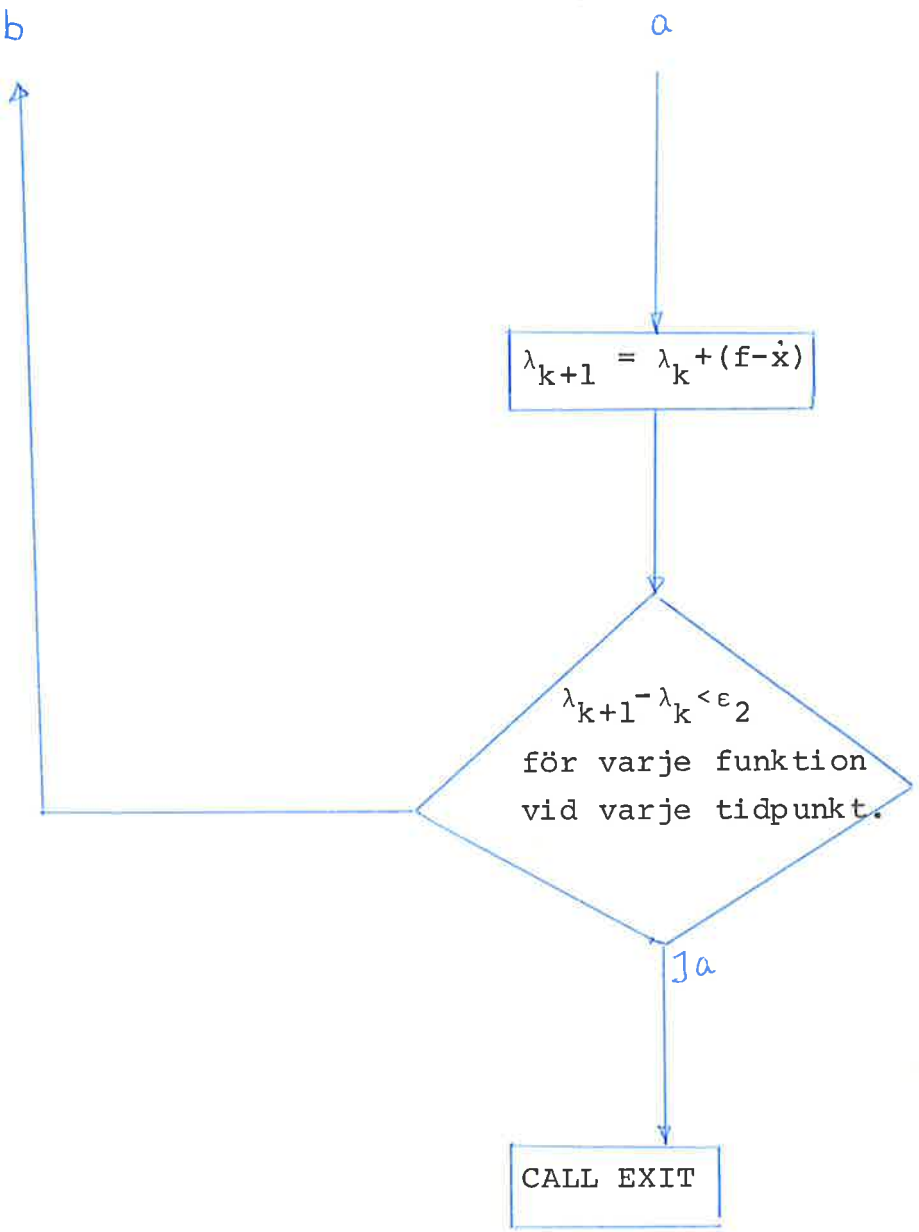
5 IBM, SSP, Algorithm QSF.

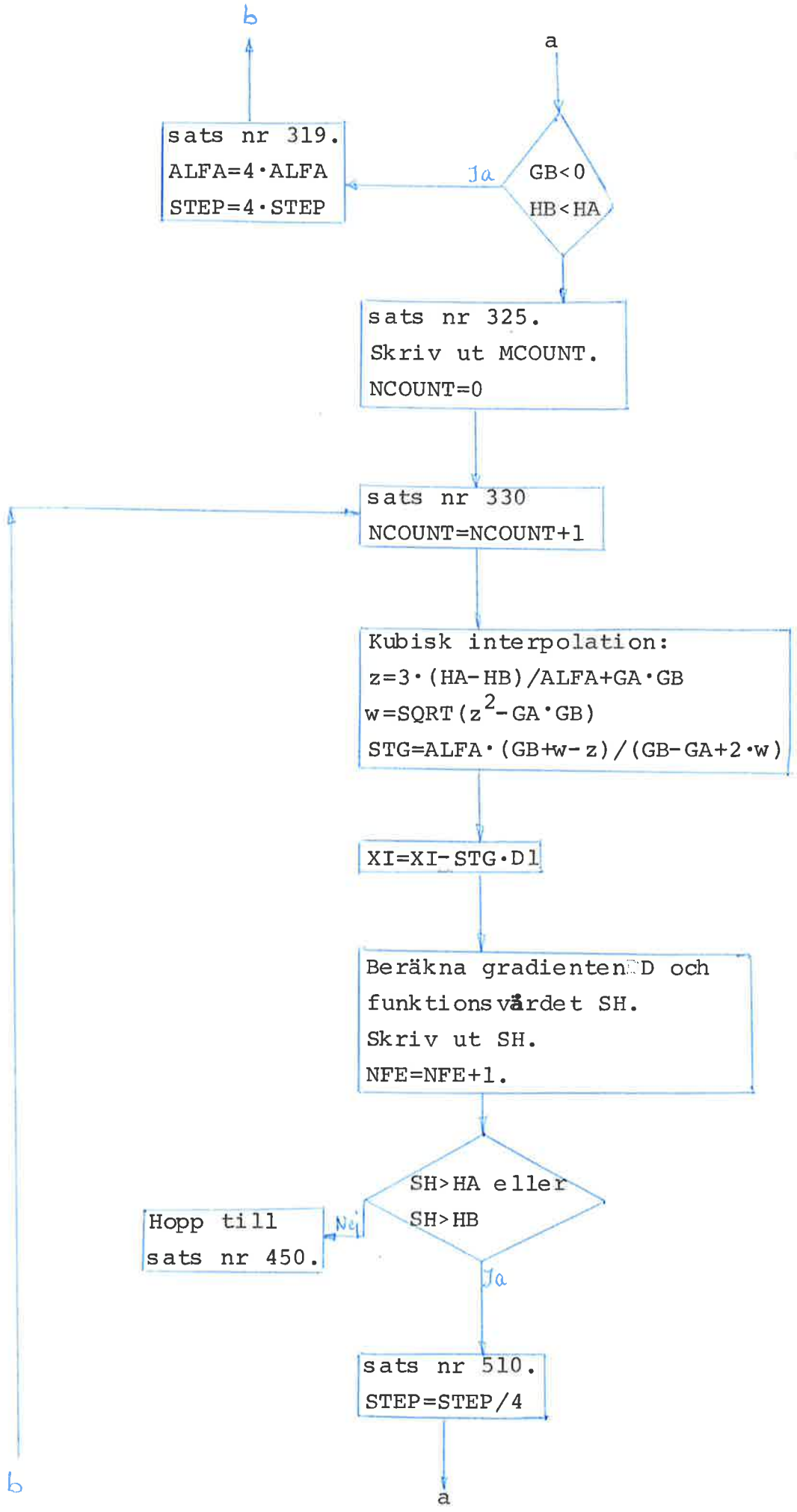
17. Appendix A. Flödesschema för Fletcher-Reeves algoritm.

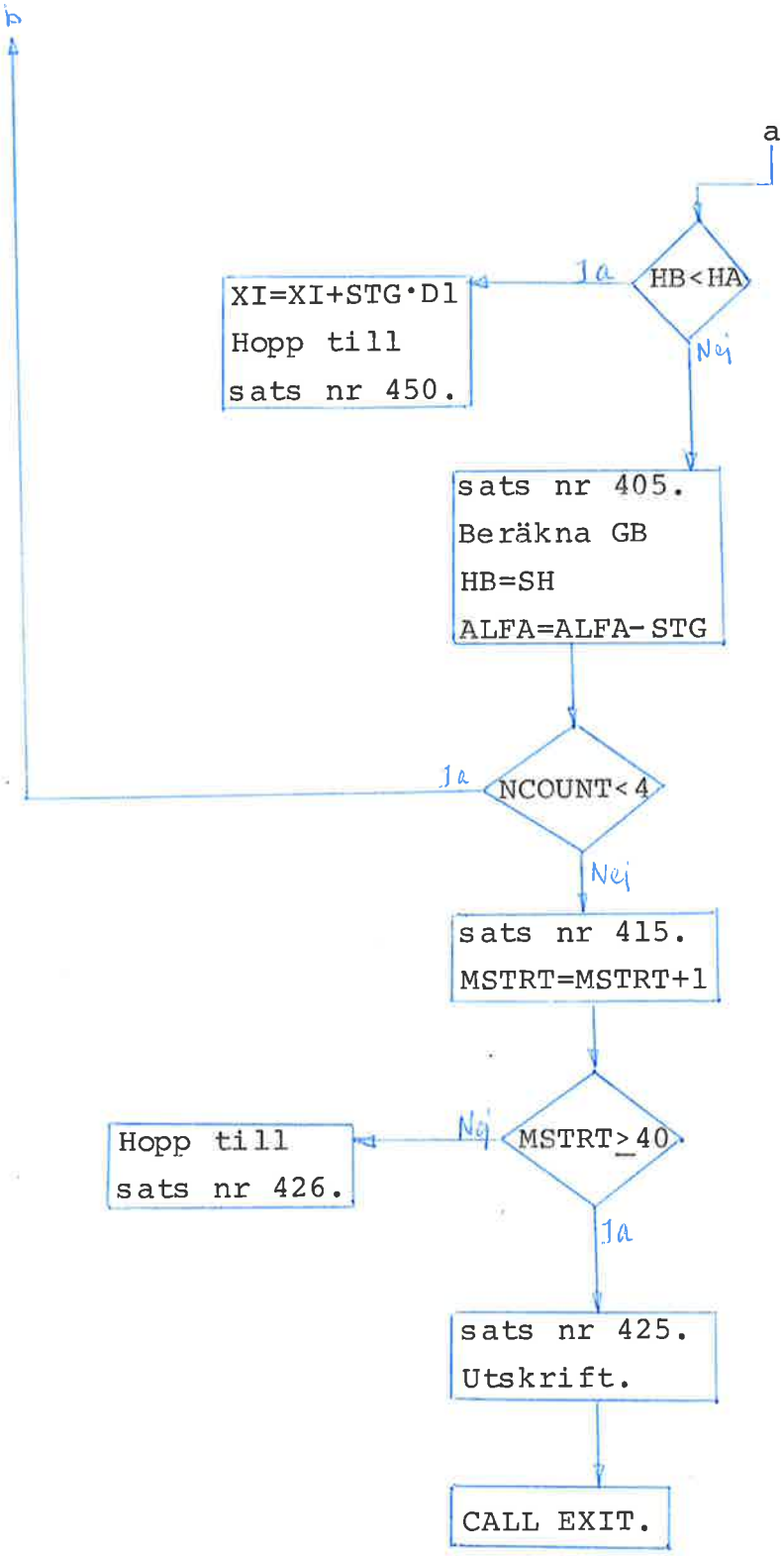


18. Appendix B. Flödesschema för den endimensionella minimeringen.




b
a
$\lambda_{k+1} = \lambda_k + (f-\dot{x})$
$\lambda_{k+1} - \lambda_k < \varepsilon_2$
för varje funktion
vid varje tidpunkt.
Ja
CALL EXIT


b
a
sats nr 319.
ALFA=4·ALFA
STEP=4·STEP
Ja
GB<0
HB<HA
sats nr 325.
Skriv ut MCOUNT.
NCOUNT=0
sats nr 330
NCOUNT=NCOUNT+1
Kubisk interpolation:
z=3·(HA-HB)/ALFA+GA·GB
w=SQRT(z2-GA·GB)
STG=ALFA·(GB+w-z)/(GB-GA+2·w)
XI=XI-STG·D1
Beräkna gradienten D och
funktionsvärdet SH.
Skriv ut SH.
NFE=NFE+1.
SH>HA eller
SH>HB
Nej
Hopp till
sats nr 450.
Ja
sats nr 510.
STEP=STEP/4
b
a


b
a
HB<HA
Ja
XI=XI+STG·D1
Hopp till
sats nr 450.
Nej
sats nr 405.
Beräkna GB
HB=SH
ALFA=ALFA-STG
NCOUNT<4
Ja
Nej
sats nr 415.
MSTRT=MSTRT+1
MSTRT≥40
Nej
Hopp till
sats nr 426.
Ja
sats nr 425.
Utskrift.
CALL EXIT.

19. Appendix C.

```
1*            SUBROUTINE USER
2*            DIMENSION XIN(6),UNOM(2,501),XPNOM(6,501),X(6),U(2),F(6),FX(6,6),
3*           *FU(6,2),SFX(6),SLX(6),SLU(2),SLAMDA(6,501)
4*            COMMON/USER1/N,NU,NSTEP,H,T0,T1,C,NPRINT,XIN,UNOM,STEP,
5*           *XPNOM,SLAMDA,X,U,EPS1,EPS2,T,EPS3
6*            COMMON/USER2/SF,SL,SFX,FX,FU,SLX,SLU,F
7*            ENTRY INIT
8*            STEP=0.01
9*            EPS1=0.35
10*           EPS2=0.01
11*           EPS3=5.
12*           N=2
13*           NU=1
14*           NSTEP=500
15*           H=0.01
16*           NPRINT=50
17*           T0=0.
18*           T1=5.
19*           C=1.
20*           XIN(1)=0.
21*           XIN(2)=1.
22*           UNOM(1,1)=0.
23*           XPNOM(1,1)=0.
24*           XPNOM(2,1)=0.
25*           SLAMDA(1,1)=0.
26*           SLAMDA(2,1)=0.
27*           DO 1 I=1,NSTEP
28*           UNOM(1,I+1)=0.
29*           XPNOM(1,1+I)=0.
30*           XPNOM(2,1+I)=0.
31*           SLAMDA(1,1+I)=0.
32*     1     SLAMDA(2,1+I)=0.
33*           RETURN
34*     C
35*           ENTRY FINLOS
36*           SF=0.
37*           RETURN
38*     C
39*           ENTRY BBOUND
40*           SFX(1)=0.
41*           SFX(2)=0.
42*            RETURN
43*     C
44*           ENTRY PDER
45*           FX(1,1)=1-X(2)**2
46*           FX(1,2)=-2.*X(2)*X(1)-1
47*           FX(2,1)=1.
48*           FX(2,2)=0.
49*           FU(1,1)=1.
50*           FU(2,1)=0.
51*           SLX(1)=2.*X(1)
52*           SLX(2)=2.*X(2)
53*           SLU(1)=2.*U(1)
54*           RETURN
55*     C
56*           ENTRY INTLOS
57*           SL=X(1)**2+X(2)**2+U(1)**2
58*           RETURN
59*     C
60*           ENTRY SYSTEM
61*           F(1)=(1-X(2)**2)*X(1)-X(2)+U(1)
62*           F(2)=X(1)
63*           RETURN
64*     C
65*           END
```

20. Appendix D. Listning av program IMF1:

```
 1*            DIMENSION XIN(6),UNOM(2,501),XNOM(6,501),SLAMDA(6,501),SFX(
 2*           *6),FX(6,6),XPNOM(6,501),FU
 3*           *(6,2),SLX(6),SLU(2),
 4*           *XVEC(6),UVEC(2),F(6),BETA1(501),E(501),XI(8,501),
 5*           *D(8,501),D1(8,501),ETA(501),SINT(501),A(8,501)
 6*            COMMON/USER1/N,NU,NSTEP,H,T0,T1,C,NPRINT,XIN,UNOM,STEP,
 7*           *XPNOM,SLAMDA,XVEC,UVEC,EPS1,EPS2,T,EPS3
 8*            COMMON/USER2/SF,SL,SFX,FX,FU,SLX,SLU,F
 9*      C
10*      C     INITIAL PRINTOUTS
11*      C
12*            CALL INIT
13*            PRINT 2
14*      2     FORMAT(36H RESULTS AFTER MINIMISING A FUNCTION,/,
15*           *59H J=SF(X(TF))+INT(SL(X,U)) UNDER THE CONDITION DX/DT=F(X,U),
16*           */,44H WHERE X=(X1,X2,..,XN) AND U=(U1,U2,...,U7).,/,
17*           *35H TF IS THE TIME OF THE FINAL STATE.,/,/,
18*           *16H INITIAL VALUES:,/)
19*            PRINT 4,N,NU,NSTEP,H
20*      4     FORMAT(3H N=,I1,5H  NU=,I1,8H  NSTEP=,I3,4H  H=,F5.3)
21*            PRINT 6,T0,T1,NPRINT
22*      6     FORMAT(4H T0=,F4.2,5H  T1=,F4.2,9H  NPRINT=,I3)
23*            PRINT 7,C,EPS1,EPS2,STEP
24*      7     FORMAT(3H C=,F4.1,6H EPS1=,F7.3,6H EPS2=,F7.5)
25*            PRINT 8
26*      8     FORMAT(5X,1HT,3X,4HUNOM)
27*            NSTP=NSTEP+1
28*            IP=NPRINT
29*            DO 30 I=1,NSTP
30*            IF (IP-NPRINT) 30,20,20
31*      20    T=T0+(I-1)*H
32*            IP=0
33*            PRINT 25,T,(UNOM(K,I),K=1,NU)
34*      25    FORMAT(F12.3,2(F12.3))
35*      30    IP=IP+1
36*            PRINT 10
37*      10    FORMAT(5X,1HT,6X,5HXPNOM)
38*            IP=NPRINT
39*            DO 50 I=1,NSTP
40*            IF (IP-NPRINT) 50,40,40
41*      40    T=T0+(I-1)*H
42*            IP=0
43*            PRINT 45,T,(XPNOM(K,I),K=1,N)
44*      45    FORMAT(F12.3,6(F12.3))
```

```
      50    IP=IP+1
            PRINT 12
      12    FORMAT(5X,1HT,6X,6HSLAMDA)
            IP=NPRINT
            DO 70 I=1,NSTP
            IF (IP-NPRINT) 70,60,60
      60    T=T0+(I-1)*H
            IP=0
            PRINT 65,T,(SLAMDA(K,I),K=1,N)
      65    FORMAT(F12.3,6(F12.3))
      70    IP=IP+1
      C
            BETA1(NSTP)=0
            ASTEP=STEP
      C
      C     XNOM COMPUTED FROM XPNOM
      C
            DO 80 K=1,N
            DO 75 I=1,NSTP
      75    E(I)=XPNOM(K,I)
            CALL QSF(H,E,SINT,NSTP)
            DO 80 I=1,NSTP
      80    XNOM(K,I)=SINT(I)+XIN(K)
      C
      C     PRINT XNOM
      C
            PRINT 14
      14    FORMAT(26H XNOM COMPUTED FROM XPNOM:)
            PRINT 16
      16    FORMAT(5X,1HT,6X,4HXNOM)
            IP=NPRINT
            DO 100 I=1,NSTP
            IF (IP-NPRINT) 100,90,90
      90     IP=0
            T=T0+(I-1)*H
            PRINT 95,T,(XNOM(J,I),J=1,N)
      95    FORMAT(F12.3,6(F12.3))
      100   IP=IP+1
      C
            NXI=N+NU
      C
      C     XI IS MADE FROM XPNOM AND UNOM
      C
            DO 110 I=1,NSTP
            DO 105 J=1,N
      105   XI(J,I)=XPNOM(J,I)
            DO 110 J=1,NU
            NK=J+N
      110   XI(NK,I)=UNOM(J,I)
      C
      C     MSTRT COUNTS HOW MANY TIMES THE INTERPOLATION FAILS
      C
      150   MSTRT=0
            IGRAD=1
      C
      C     JUMP TO COMPUTATION OF GRAD
      C
            GO TO 450
      154   IGRAD=0
      C
      C     FIRST SEARCHDIRECTION
      C
            DO 155 I=1,NSTP
            DO 155 K=1,NXI
```

```
G
109*      155   D1(K,I)=-D(K,I)
110*      C
111*      C     START MINIMISING
112*      C
113*      200   NFE=0
114*            PRINT 190,STEP
115*      190   FORMAT(6H STEP=,F9.4,/)
116*      C
117*      C     NFE COUNTS HOW MANY TIMES THE FUNCTION HAS BEEN AVALUATED
118*      C
119*            IFAIL=0
120*            NFE=NFE+1
121*      C
122*      C     COMPUTE SH
123*      C
124*            DO 220 I=1,NSTP
125*            T=T0+(I-1)*H
126*            DO 205 J=1,N
127*      205   XVEC(J)=XNOM(J,I)
128*            DO 210 JU=1,NU
129*            JK=JU+N
130*      210   UVEC(JU)=XI(JK,I)
131*            CALL INTLOS
132*            CALL SYSTEM
133*            P=0.
134*            DO 215 K=1,N
135*      215   P=SLAMDA(K,I)*(F(K)-XI(K,I))+P+C*(XI(K,I)-F(K))**2
136*      220   E(I)=P+SL
137*            CALL QSF(H,E,SINT,NSTP)
138*            CALL FINLOS
139*            SH=SINT(NSTP)+SF
140*      C
141*      C     STOP COMPUTE SH
142*      C
143*      C     PRINT SH
144*      C
145*            PRINT 224
146*      224   FORMAT(37H COMPUTED VALUES DURING MINIMISATION:)
147*            PRINT 225,SH
148*      225   FORMAT(4H SH=,F7.3)
149*            HB=SH
150*      C
151*      C     COMPUTE GB
152*      C
153*            DO 230 I=1,NSTP
154*            E(I)=0.
155*            DO 230 K=1,NXI
156*      230   E(I)=D(K,I)*D1(K,I)+E(I)
157*            CALL QSF(H,E,SINT,NSTP)
158*            GB=SINT(NSTP)
159*            PRINT 231,GB
160*      231   FORMAT(19H FIRST VALUE OF GB:,/,4H GB=,F11.3/)
161*            IF (GB) 240,235,235
162*      C
163*      C     IF GB>0 THEN GB=-GB AND SEARCHDIR=-SEARCHDIR
164*      C
165*      235   GB=-GB
166*            DO 236 I=1,NSTP
167*            DO 236 K=1,NXI
168*      236   D1(K,I)=-D1(K,I)
169*      240   ALFA=STEP
170*            MCOUNT=0
171*      C
172*      C     START EXTRAPOLATING
```

```
G

173*      C
174*      250   HA=HB
175*            GA=GB
176*            MCOUNT=MCOUNT+1
177*      C
178*      C     COMPUTE NEW XI
179*      C
180*      254   DO 255 I=1,NSTP
181*            DO 255 K=1,NXI
182*      255   XI(K,I)=XI(K,I)+ALFA*D1(K,I)
183*      C
184*      C     XNOM COMPUTED FROM XPNOM
185*      C
186*      256    DO 280 K=1,N
187*            DO 275 I=1,NSTP
188*      275   E(I)=XI(K,I)
189*            CALL QSF(H,E,SINT,NSTP)
190*            DO 280 I=1,NSTP
191*      280   XNOM(K,I)=SINT(I)+XIN(K)
192*      C
193*      C     COMPUTE AND PRINT NEW SH
194*      C
195*            DO 300 I=1,NSTP
196*            T=T0+(I-1)*H
197*            DO 285 J=1,N
198*      285   XVEC(J)=XNOM(J,I)
199*            DO 290 JU=1,NU
200*            JK=JU+N
201*      290   UVEC(JU)=XI(JK,I)
202*            CALL INTLOS
203*            CALL SYSTEM
204*            P=0.
205*            DO 295 K=1,N
206*      295   P=SLAMDA(K,I)*(F(K)-XI(K,I))+P+C*(XI(K,I)-F(K))**2
207*      300   E(I)=P+SL
208*            CALL QSF(H,E,SINT,NSTP)
209*            CALL FINLOS
210*            SH=SINT(NSTP)+SF
211*            PRINT 305,SH
212*      305   FORMAT(/,/,37H SH COMPUTED AFTER EXTRAPOLATION: SH=,F12.3)
213*      C
214*            NFE=NFE+1
215*            IGRAD=2
216*      C
217*      C     JUMP TO COMPUTATION OF GRAD
218*      C
219*            GO TO 450
220*      310   IGRAD=0
221*            HB=SH
222*      C
223*      C     COMPUTE GB
224*      C
225*            DO 315 I=1,NSTP
226*            E(I)=0.
227*            DO 315 K=1,NXI
228*      315   E(I)=D(K,I)*D1(K,I)+E(I)
229*            CALL QSF(H,E,SINT,NSTP)
230*            GB=SINT(NSTP)
231*            PRINT 317,GB
232*      317   FORMAT(/,/,44H GB COMPUTED AFTER EXTRAPOLATION NEW XI: GB=,F12.3)
233*            IF ((GB.LT.0.).AND.(HB.GT.HA)) GO TO 254
234*      C
235*      C     IF GB>0 OR HB>HA THEN START INTERPOLATING
236*      C
```

```
G
237*            IF ((GB.GE.0.).OR.(HB.GT.HA)) GO TO 325
238*      319   ALFA=ALFA*(GA-GB)/(4.*GA)+4.*ALFA*GB/GA
239*            STEP=STEP*(GA-GB)/(4.*GA)+4.*STEP*GB/GA
240*            PRINT  318,ALFA,STEP,GA,GB
241*      318   FORMAT(6H ALFA=,F7.4,6H STEP=,F7.4,4H GA=,F12.8,4H GB=,F12.8)
242*      C
243*      C     IF MCOUNT > 6 CALL EXIT ELSE JUMP BACK TO EXTRAPOLATION
244*      C
245*            IF (MCOUNT-6) 250,250,316
246*      316   PRINT 320
247*      320   FORMAT(1X,14HMCOUNT .GT. 4.)
248*            STEP=ASTEP
249*            CALL EXIT
250*      325   NCOUNT=0
251*      C
252*      C     STOP EXTRAPOLATION
253*      C
254*      C     START INTERPOLATION
255*      C
256*            ASTEP=STEP
257*            PRINT 324,MCOUNT
258*      324   FORMAT(/,8H MCOUNT=,I2)
259*            PRINT 329
260*      329   FORMAT(20H START INTERPOLATION,/,/)
261*      330   NCOUNT=NCOUNT+1
262*            Z=3.*(HA-HB)/ALFA+GA+GB
263*            O=Z**2-GA*GB
264*            IF (O) 332,333,333
265*      332   O=0
266*      333   W=SQRT(O)
267*            STG=ALFA*(GB+W-Z)/(GB-GA+2.*W)
268*            PRINT 331,STG
269*      331   FORMAT(37H STEPLENGTH WHILE INTERPOLATING: STG=,F12.8,/)
270*      C
271*      C     STG IS THE STEPLENGTH WHILE INTERPOLATING
272*      C
273*      C     COMPUTE NEW XI
274*      C
275*            DO 340 I=1,NSTP
276*            DO 340 K=1,NXI
277*      340   XI(K,I)=XI(K,I)-STG*D1(K,1)
278*      C
279*      C     UTSKRIFT AV XI
280*            PRINT 115
281*      115   FORMAT(4H XI=)
282*            IP=NPRINT
283*            DO 130 I=1,NSTP
284*            IF (IP-NPRINT) 130,120,120
285*      120   IP=0
286*            T=T0+(I-1)*H
287*            PRINT 125,T,(XI(J,I),J=1,NXI)
288*      125   FORMAT(F7.3,8(F7.3))
289*      130   IP=IP+1
290*      C
291*      C     XNOM COMPUTED FROM XPNOM
292*      C
293*            DO 350 K=1,N
294*            DO 345 I=1,NSTP
295*      345   E(I)=XI(K,I)
296*            CALL QSF(H,E,SINT,NSTP)
297*            DO 350 I=1,NSTP
298*      350   XNOM(K,I)=SINT(I)+XIN(K)
299*      C
300*      C     COMPUTE NEW SH
```

```
G

301*      C
302*            DO 370 I=1,NSTP
303*            T=T0+(I-1)*H
304*            DO 355 J=1,N
305*      355   XVEC(J)=XNOM(J,I)
306*            DO 360 JU=1,NU
307*            JK=JU+N
308*      360   UVEC(JU)=XI(JK,I)
309*            CALL SYSTEM
310*            CALL INTLOS
311*            P=0.
312*            DO 365 K=1,N
313*      365   P=SLAMDA(K,I)*(F(K)-XI(K,I))+P+C*(XI(K,I)-F(K))**2
314*      370   E(I)=P+SL
315*            CALL QSF(H,E,SINT,NSTP)
316*            CALL FINLOS
317*            SH=SINT(NSTP)+SF
318*            PRINT 371,SH
319*      371   FORMAT(/,/,37H SH COMPUTED WHILE INTERPOLATING: SH=,F15.6,/,/)
320*            NFE=NFE+1
321*            IGRAD=3
322*      C
323*      C     JUMP TO COMPUTATION OF GRAD
324*      C
325*            GO TO 450
326*      375   IGRAD=0
327*      C
328*      C     COMPUTE GB
329*      C
330*            DO 372 I=1,NSTP
331*            E(I)=0.
332*            DO 372 K=1,NXI
333*      372   E(I)=D(K,I)*D1(K,I)+E(I)
334*            CALL QSF(H,E,SINT,NSTP)
335*            GB=SINT(NSTP)
336*            PRINT 376,SH,HA,HB,GB
337*      376   FORMAT(18H MINIMUM FOUND?SH=,F15.6,4H HA=,F15.6,4H HB=,F15.6,/,
338*           *4H GB=,F12.8)
339*      C
340*      C     IF HA>SH AND HB>SH THEN MINIMUM IS FOUND
341*      C
342*            IF (SH.LE.HA.AND.SH.LE.HB) GO TO 400
343*            STEP=STEP/4
344*            IF (HB-HA) 380,405,405
345*      380   DO 385 I=1,NSTP
346*            DO 385 K=1,NXI
347*      385   XI(K,I)=XI(K,I)+STG*D1(K,I)
348*      C
349*      C     XNOM COMPUTED FROM XPNOM
350*      C
351*            DO 395 K=1,N
352*            DO 390 I=1,NSTP
353*      390   E(I)=XI(K,I)
354*            CALL QSF(H,E,SINT,NSTP)
355*            DO 395 I=1,NSTP
356*      395   XNOM(K,I)=SINT(I)+XIN(K)
357*      C
358*      400   IFAIL=0
359*      C
360*      C     MINIMUM IS FOUND . JUMP TO COMPUTATION OF GRAD
361*            GO TO 450
362*      C
363*      C     COMPUTE NEW GB
364*      C
```

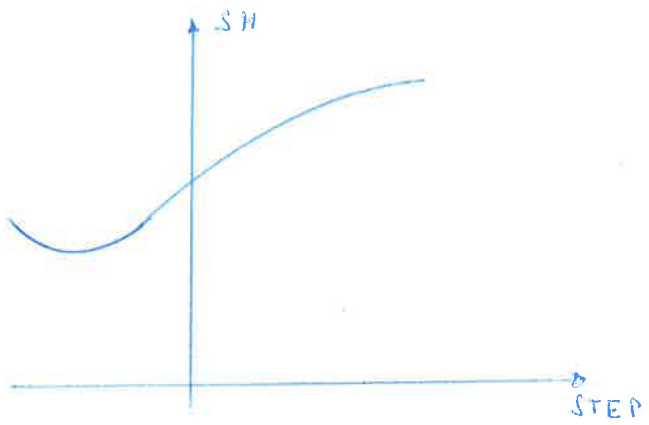
```
G
365*      405   DO 410 I=1,NSTP
366*            E(I)=0.
367*            DO 410 K=1,NXI
368*      410   E(I)=E(I)+D(K,I)*D1(K,I)
369*            CALL QSF(H,E,SINT,NSTP)
370*            GB=SINT(NSTP)
371*      C
372*            HB=SH
373*            ALFA=ALFA-STG
374*            IF (NCOUNT-4) 330,415,415
375*      C
376*      C     IF INTERPOLATION FAILS THEN RESTART WITH NEGATIVE
377*      C     GRAD AS SEARCHDIRECTION
378*      C
379*      415   IGRAD=1
380*            STEP=ASTEP
381*            IL=0
382*            MSTRT=MSTRT+1
383*      C
384*      C      IF MORE THAN 4 RESTARTS ARE REQUIRED FOR A GIVEN
385*      C     SLAMDA CALL EXIT
386*      C
387*            IF (MSTRT-4) 426,426,420
388*      420   PRINT 425,MSTRT
389*      425   FORMAT(1X,25H TOO MANY RESTARTS MSTRT=,I3)
390*            CALL EXIT
391*      426   CONTINUE
392*      C
393*      C     STOP MINIMISING
394*      C
395*      C     START COMPUTATION OF GRAD
396*      C
397*      C     GRAD WITH RESPECT TO XPNOM
398*      C
399*      450   T=T1
400*            I=NSTP
401*            DO 430 J=1,N
402*      430   XVEC(J)=XNOM(J,I)
403*            DO 435 JU=1,NU
404*            JK=JU+N
405*      435   UVEC(JU)=XI(JK,I)
406*            CALL BBOUND
407*            DO 480 KP=1,N
408*            DO 466 I=1,NSTP
409*            T=T0+(I-1)*H
410*            DO 455 J=1,N
411*      455   XVEC(J)=XNOM(J,I)
412*            DO 460 JU=1,NU
413*            JK=JU+N
414*      460   UVEC(JU)=XI(JK,I)
415*            CALL SYSTEM
416*            CALL PDER
417*            IK=NSTP-I+1
418*            ETA(IK)=0.
419*            DO 465 K=1,N
420*      C
421*      C     BACKWORDS INTEGRATION
422*      C
423*      465   ETA(IK)=ETA(IK)+FX(K,KP)*SLAMDA(K,I)+2.*C*FX(K,KP)*(F(K)-XI(K,I))
424*      466   ETA(IK)=ETA(IK)+SLX(KP)
425*            CALL QSF(H,ETA,SINT,NSTP)
426*            DO 480 I=1,NSTP
427*            T=T0+(I-1)*H
428*            DO 470 J=1,N
```

```
IG
429*      470   XVEC(J)=XNOM(J,I)
430*            DO 475 JU=1,NU
431*            JK=JU+N
432*      475   UVEC(JU)=XI(JK,I)
433*            CALL SYSTEM
434*            IK=NSTP-I+1
435*      480   D(KP,I)=SINT(IK)-SLAMDA(KP,I)-2.*C*(F(KP)-XI(KP,I))+SFX(KP)
436*      C
437*      C     GRAD WITH RESPECT TO UNOM
438*      C
439*            DO 500 I=1,NSTP
440*            T=T0+(I-1)*H
441*            DO 485 JU=1,NU
442*            JK=JU+N
443*      485   UVEC(JU)=XI(JK,I)
444*            DO 490 J=1,N
445*      490   XVEC(J)=XNOM(J,I)
446*            CALL PDER
447*            CALL SYSTEM
448*            DO 500 KU=1,NU
449*            KN=KU+N
450*            D(KN,I)=0.
451*            DO 495 K=1,N
452*      495   D(KN,I)=D(KN,I)+FU(K,KU)*SLAMDA(K,I)+2.*C*FU(K,KU)*(F(K)-XI(K,I))
453*      500   D(KN,I)=D(KN,I)+SLU(KU)
454*      C
455*      C     STOP COMPUTE GRAD
456*      C
457*      C     JUMP BACK:S
458*      C
459*            IF (IGRAD) 501,506,501
460*      501   CONTINUE
461*            GO TO (555,310,375),IGRAD
462*      C
463*      C      PRINTOUTS OF VALUES COMPUTED DURING MINIMISATION
464*      C
465*      506   CONTINUE
466*            PRINT 515
467*      515   FORMAT(/,24H VALUES AFTER MINIMISING,/)
468*            PRINT 505,NFE,NCOUNT,STEP
469*      505   FORMAT(5H NFE=,I1,8H NCOUNT=,I2,7H  STEP=,F12.8)
470*      C
471*            BETA2=BETA1(NSTP)
472*      C
473*      C     COMPUTE GRAD SQUARE
474*      C
475*      555   DO 560 I=1,NSTP
476*            E(I)=0.
477*            DO 560 K=1,NXI
478*      560   E(I)=D(K,I)*D(K,I)+E(I)
479*            CALL QSF(H,E,BETA1,NSTP)
480*            PRINT 565,BETA1(NSTP)
481*            IF (IGRAD-1) 570,154,570
482*      565   FORMAT(7H BETA1=,F12.5)
483*      570   BETA=BETA1(NSTP)/BETA2
484*            IF (BETA.GT.0.999.AND.BETA.LT.1.001) GO TO 605
485*            PRINT 575,BETA
486*      575   FORMAT(6H BETA=,F7.3,/,/,/,/)
487*      C
488*      C      IF BETA IS NOT SMALL ENOUGH THEN COMPUTE NEW SLAMDA
489*      C     AND START FROM THE BEGINNING. ELSE NEW SEARCHDIRECTION
490*      C
491*            IF (BETA1(NSTP)-EPS1) 605,580,580
492*      580   DO 585 I=1,NSTP
```

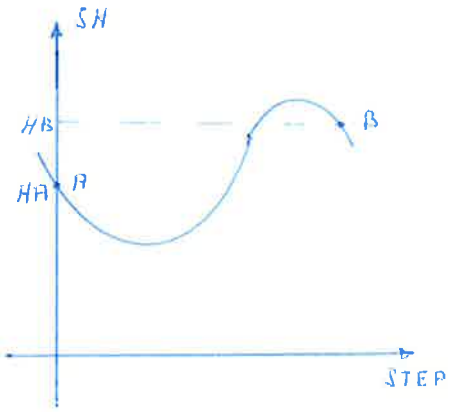
```
      DO 585 K=1,NXI
  585 D1(K,I)=BETA*D1(K,I)-D(K,I)
      IL=IL+1
      IF (IL-5) 590,415,415
  590 CONTINUE
      GO TO 200
C
C     COMPUTE NEW SLAMDA
C
  605 DO 620 I=1,NSTP
      T=T0+(I-1)*H
      DO 610 K=1,N
  610 XVEC(K)=XNOM(K,I)
      DO 615 KU=1,NU
      KK=KU+N
  615 UVEC(KU)=XI(KK,I)
      CALL SYSTEM
      DO 620 K=1,N
      SLAMDA(K,I)=SLAMDA(K,I)+2.*C*(F(K)-XI(K,I))
  620 A(K,I)=XI(K,I)-F(K)
      PRINT 510
      EPS1=EPS1/3.
  510 FORMAT(6X,1HT,15X,2HXI)
      IP=NPRINT
      DO 535 I=1,NSTP
      IF (IP-NPRINT) 535,520,520
  520 IP=0
      T=T0+(I-1)*H
      PRINT 525,T,(XI(K,I),K=1,NXI)
  525 FORMAT(F12.3,8(F12.3))
  535 IP=IP+1
      PRINT 530
  530 FORMAT(6X,1HT,15X,4HXNOM)
      IP=NPRINT
      DO 550 I=1,NSTP
      IF (IP-NPRINT) 550,540,540
  540 IP=0
      T=T0+(I-1)*H
      PRINT 545,T,(XNOM(K,I),K=1,N)
  545 FORMAT(F12.3,6(F12.3))
  550 IP=IP+1
C
C     IF A(K,I) IS SMALL ENOUGH THEN GO ON TESTING
C     ELSE RESTART THE PROGRAM
C
      PRINT 625
  625 FORMAT(8H SLAMDA=)
      IP=NPRINT
      DO 640 I=1,NSTP
      IF (IP-NPRINT) 640,630,630
  630 IP=0
      T=T0+(I-1)*H
      PRINT 635,T,(SLAMDA(K,I),K=1,N)
  635 FORMAT(F12.3,6(F12.3))
  640 IP=IP+1
      DO 650 I=1,NSTP
      DO 650 K=1,N
      P=A(K,I)
      IF (P) 641,642,642
  641 P=-P
  642 IF (P-EPS2) 650,645,645
  645 GO TO 150
  650 CONTINUE
      CALL EXIT
```

```
ING

  557*            END

END OF COMPILATION:          NO  DIAGNOSTICS.
```

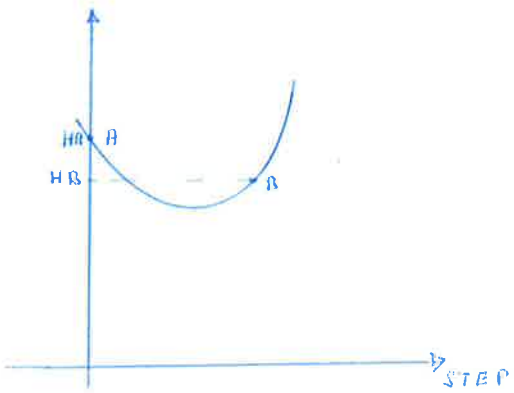
21. Figurbilaga.



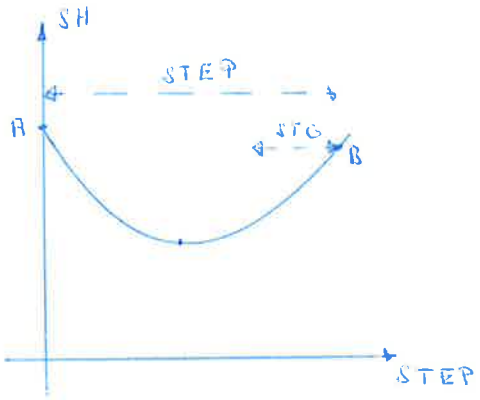
Figur 1.



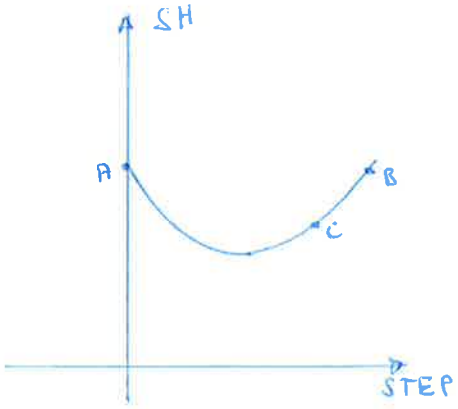
Figur 2a.



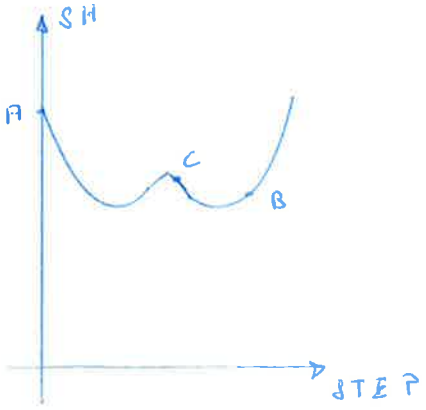
Figur 2b.



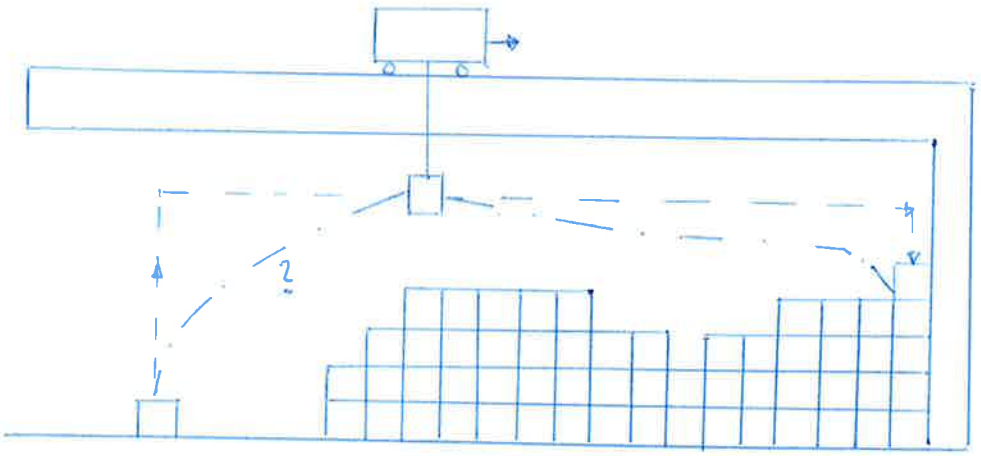
Figur 3.



Figur 4a.



Figur 4b.



Figur 5.