KONJUGERADE GRADIENTMETODEN FÖR OPTIMALA STYRPROBLEM

LEIF PERSSON

RE-143 Juli 1974
Inst. för Reglerteknik
Lunds Tekniska Högskola

KONJUGERADE GRADIENTMETODEN FÖR OPTIMALA STYRPROBLEM.

LEIF PERSSON

RE-143 juli 1974
Inst. för Reglerteknik
Lunds Tekniska Högskola

KONJUGERADE GRADIENTMETODEN FÖR OPTIMALA STYRPROBLEM.

Examensarbete vid Institutionen för Reglerteknik,
Lunds Tekniska Högskola.
Utfört under vårterminen 1974 av Leif Persson
Handledare: Forskarassistent Krister Mårtensson

ABSTRACT.

In this thesis a computer program (ICG1) for numerical of optimal control problems is presented.
The program is based on a conjugate gradient method in the control space, and at every computation of the gradient, the system equations are solved.
The program is written in FORTRAN and to solve different examples on a computer, only one subroutine USER needs to be changed.
The program has been tested on examples of different complexity and comparisons with two other methods ([1], [2]) have been done.

I detta examensabete presenteras ett datorprogram (ICG1) för numerisk lösning av optimala styrproblem.
Programmet är baserat på en konjugerad gradientmetod i insignalrummet, och varje beräkning av gradienten innebär en lösning av systemekvationerna.
Programmet är skrivet i FORTRAN och för att köra olika exempel på dator, behöver endast en subrutin USER ändras.
Programmet har testats på exempel av olika svårighetsgrad och jämförelser med andra lösningsmetoder ([1],[2]) har gjorts.

2. Innehållsförteckning.

3. Problemformulering.

Givet: Ett dynamiskt system beskrivet av $\dot{x}(t)=f[x(t), u(t),t]$ med randvärde $x(t_0)=x_0$. Tillståndsvariablerna $x(t)=\begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}$ har dimensionen n, styrvariablerna $u(t)=\begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix}$ har dimensionen n .

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

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

under det att bivillkoret $\dot{x}(t)=f[x(t),u(t)]$ är uppfyllt.

Metod: Vid beräkning av gradienten integreras systemekvationerna, vilket gör, att för ett givet u , bestäms x entydigt ur dessa. Förlustfunktionen J blir därför en funktional av u och därför beräknas bara gradienten av J m.a.p. u. Förlustfunktionen J minimeras med hjälp av en modifierad Fletcher-reeves algoritm. För grundligare studier av denna metod hänvisar jag till [3].

Kommentar: Observera att $x_1, x_2, \ldots, x_n$ och $u_1, u_2 \ldots, u_n$ är funktioner av tiden. F är en funktion, som endast beror på sluttillståndet för x. Vid första Runge-Kutta integrationen används en initialgissning på u(t) från subrutinen USER.

4. Teori.

Jag använder mig av Fletcher-Reeves metod, vilken är en konjugerad gradientmetod, för att numeriskt minimera J. För att kunna använda denna metod behöver man kunna beräkna gradienten av J m.a.p. u .

Kommentar: För givet u bestäms x entydigt ur $\dot{x}=f(x,u)$, $x(t_0)=x_0$. Vi kan därför betrakta J som en funktional enbart av u.

Beräkning av gradienten av J m.a.p. u .

Vi definierar differentialen av J med inkrementet h som:

$$\delta J(u,h)=\lim_{\varepsilon\to 0} \frac{J(u+\varepsilon h)-J(u)}{\varepsilon}$$

om $\delta J(u,h)$ existerar och är linjär och kontinuerlig i h .

Beräkning av denna differential:

Eftersom x alltid satisfierar $\dot{x}=f(x,u)$ gäller att

$$J(u)=F(x(t_1))+\int_{t_0}^{t_1} L(x,u)dt=F(x(t_1))+\int_{t_0}^{t_1}\{L(x,u)+\lambda^T(t)\cdot(f(x,u)-\dot{x})\}dt$$

för alla funktioner u(t) och alla funktioner $\lambda(t)$.

Om vi väljer $\lambda(t)$ som en deriverbar funktion, kan vi partialintegrera enligt

$$\int_{t_0}^{t_1} \lambda^T(t)\dot{x}(t)dt= \lambda^T(t_1)x(t_1)- \lambda^T(t_0)x(t_0)-\int_{t_0}^{t_1}\dot{\lambda}^T(t)x(t)dt$$

Alltså är

$$J(u)=F(x(t_1))+\lambda^T(t_1)x(t_1)-\lambda^T(t_0)x(t_0)+\int_{t_0}^{t_1}\{L(x,u)+\lambda^T f+\dot{\lambda}^T x\}dt \quad (1)$$

Beräkna $J(u+\varepsilon h)$ och observera att när u ändras från u till $u+\varepsilon h$ ändras x från x till $x+\delta x$ (eftersom $\dot{x}=f(x,u)$ alltid är satisfierad).

$$J(u+\varepsilon h)=F(x(t_1))+\lambda^T(t_1)\{x(t_1)+\delta x(t_1)\}-\lambda^T(t_0)\cdot\{x(t_0)+\delta x(t_0)\}+\int_{t_0}^{t_1}\{L(x+\delta x,u+\varepsilon h)+\lambda^T f(x+\delta x,u+\varepsilon h)+\dot{\lambda}^T(x+\delta x)\}dt$$

Taylorutveckla nu L och f kring x och u. Observera att $\delta x(t_0)=0$, eftersom initialtillståndet är fixerat.

$$J(u+\varepsilon h)=F(x(t_1))+F_x(x(t_1))\delta x(t_1)+\lambda^T(t_1)\cdot\{x(t_1)+\delta x(t_1)\}$$
$$-\lambda^T(t_0)x(t_0)+\int_{t_0}^{t_1}\{L(x,u)+L_x\delta x+L_u\varepsilon h+\lambda^T f+\lambda^T f_x\delta x+\lambda^T f_u\varepsilon h+$$
$$\lambda^T\dot{x}+\lambda^T\delta\dot{x}+(\text{högre ordningens termer i } \varepsilon h \text{ och } \delta x)\}dt \quad (2)$$

Tag (2) minus (1) och dividera med ε.

$$\frac{J(u+h)-J(u)}{\varepsilon}=\frac{1}{\varepsilon}\{F_x(x(t_1))-\lambda^T(t_1)\}\delta x(t_1)+\int_{t_0}^{t_1}\{\frac{1}{\varepsilon}(L_x+$$
$$\lambda^T f_x+\dot{\lambda}^T)\delta x+(L_u+\lambda^T f_u)h+\frac{1}{\varepsilon}\cdot(\text{högre ordningens termer i } \varepsilon h$$
$$\text{och } \delta x)\}\,dt$$

Man kan nu visa att $\frac{1}{\varepsilon}\cdot(\text{högre ordningens termer i } \varepsilon h \text{ och } \delta x)\to 0$ då $\varepsilon\to 0$. Om dessutom $\lambda(t)$ väljs så att den satisfierar ekvationen

$$L_x+\lambda^T f_x+\dot{\lambda}^T=0 \quad \forall t$$

och som randvärde till denna differentialekvation väljs $\lambda(t_1)=F_x^T(x(t_1))$, fås

$$J(u,h)=\lim_{\varepsilon\to 0}\frac{J(u+\varepsilon h)-J(u)}{\varepsilon}=\int_{t_0}^{t_1}(L_u+\lambda^T f_u)h\,dt \quad (3)$$

Om $\delta J(u,h)$ kan skrivas som $J(u,h)=\langle\nabla_u J,h\rangle$, så definierar vi gradienten av J m.a.p. u som $\nabla_u J$. Genom att jämföra mad (3) fås

$$\nabla_u J=L_u^T+f_u^T\lambda$$

<u>Sammanfattning</u>: Gradienten av J(u) i "punkten" $\bar{u}(t)$ bestäms på följande sätt:

1. Bestäm $\bar{x}(t)$ genom att integrera
$\dot{x}=f(\bar{x},\bar{u}) \quad x(t_0)=x_0$
från t_0 till t_1.

2. Integrera
$\dot{\lambda}=-L_x^T-f_x^T\lambda \qquad \lambda(t_1)=F_x^T(x(t_1))$
från t_1 till t_0. L_x och f_x utvärderas längs $\bar{x}(t)$ och u(t).

3. Bestäm

$$\nabla_u J(\bar{u},t)=L_u^T(\bar{x}(t),\bar{u}(t))+f_u^T(\bar{x}(t),\bar{u}(t))\lambda(t)$$

Fletcher-Reeves metod

Fletcher-Reeves metod finns beskriven i ett flertal standardböcker om olinjär optimering. Vi refererar därför till t.ex. [4] och ger här bara en kortfattad beskrivning av algoritmen med de modifieringar , som har gjorts.

Fletcher-Reeves metod för ändligt dimensionella problem.

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

2. För $k=0,1,\ldots,n-1$ (dimensionen för x är n)

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

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

c) Om inte $k=n-1$, sätt ny sökriktning $d_{k+1}=-g_{k+1}+\alpha_k d_k$ där $\beta_k=\dfrac{g_{k+1}^T\cdot g_{k+1}}{g_k^T\cdot g_k}$.

3. Ersätt x_0 med x_n och gå tillbaka till steg 1.

Modifierad Fletcher-Reeves för styrproblemet.

1. Beräkna gradienten $g_0=\nabla J(u_0)^T$ för givet u_0 och sätt begynnelseriktning $d_0=-g_0$.

2. a) Sätt $u_{k+1}=u_k+\alpha_k d_k$,där α_k minimerar $J(u_k+\alpha d_k)$

b) Beräkna den nya gradienten $g_{k+1}=\nabla f(u_{k+1})^T$

c) Om $\langle g_{k+1},g_{k+1}\rangle=\int_{t_0}^{t_1} g_{k+1}^T g_{k+1}dt$ sätt ny sökriktning $d_{k+1}=-g_{k+1}+\beta_k d_k$,där $\beta_k=\dfrac{g_{k+1}^T\cdot g_{k+1}}{g_k^T\cdot g_k}$ .Gå till 2a).

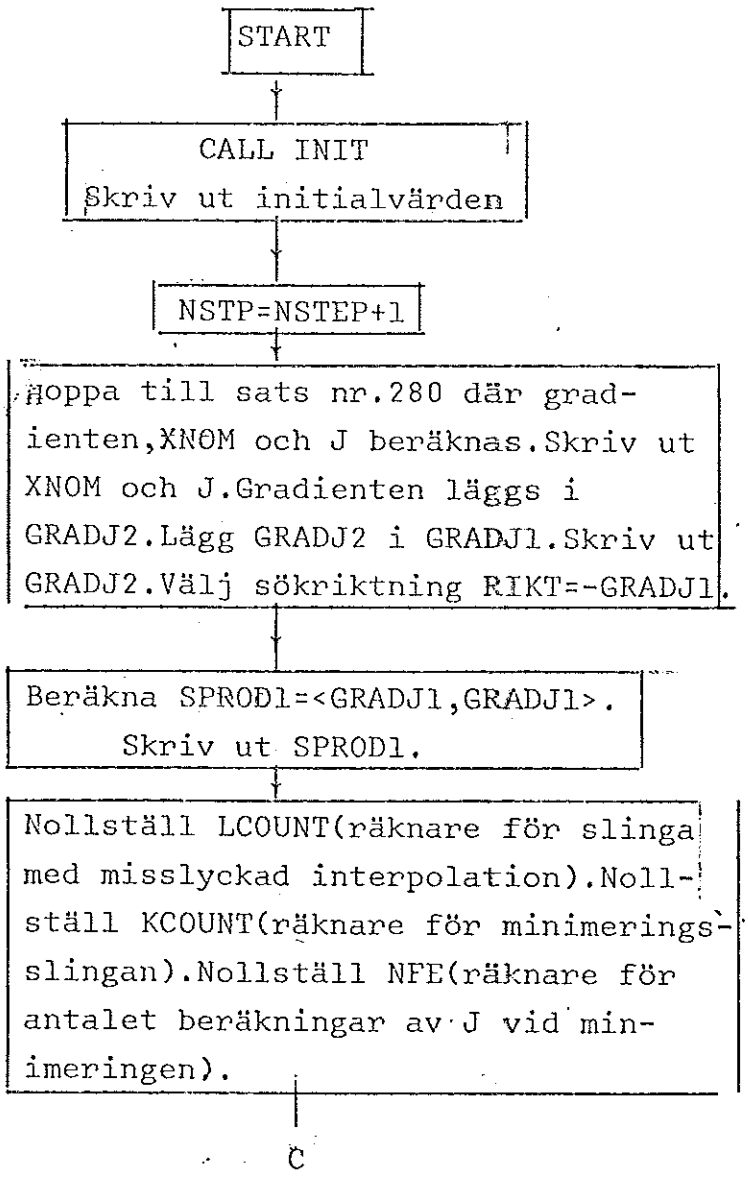
Om $\langle g_{k+1},g_{k+1}\rangle\varepsilon$ så är minimeringen färdig.

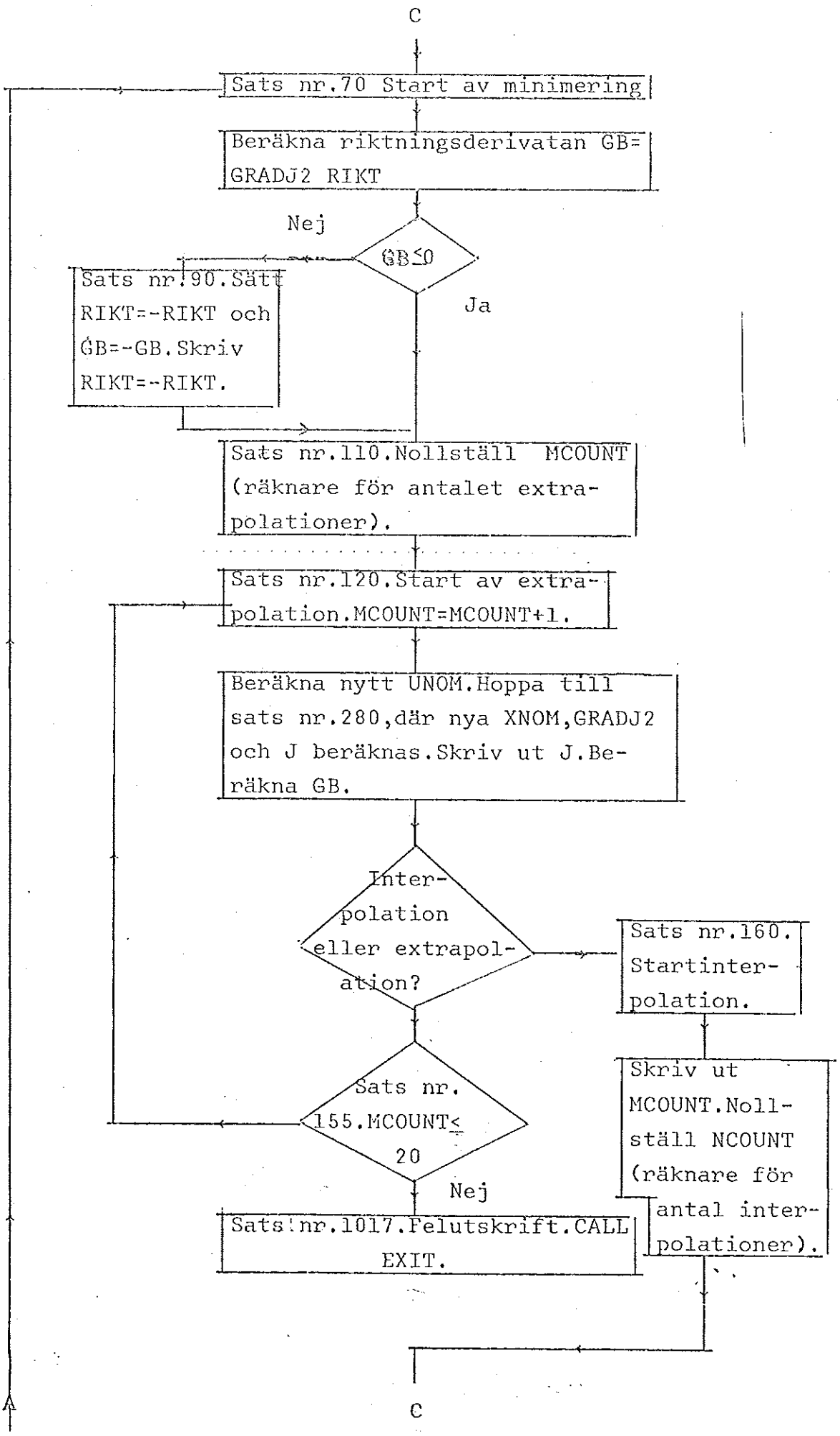
Flödesschema för den modifierade varianten av Fletcher-Reeves algoritm finns i appendix A.

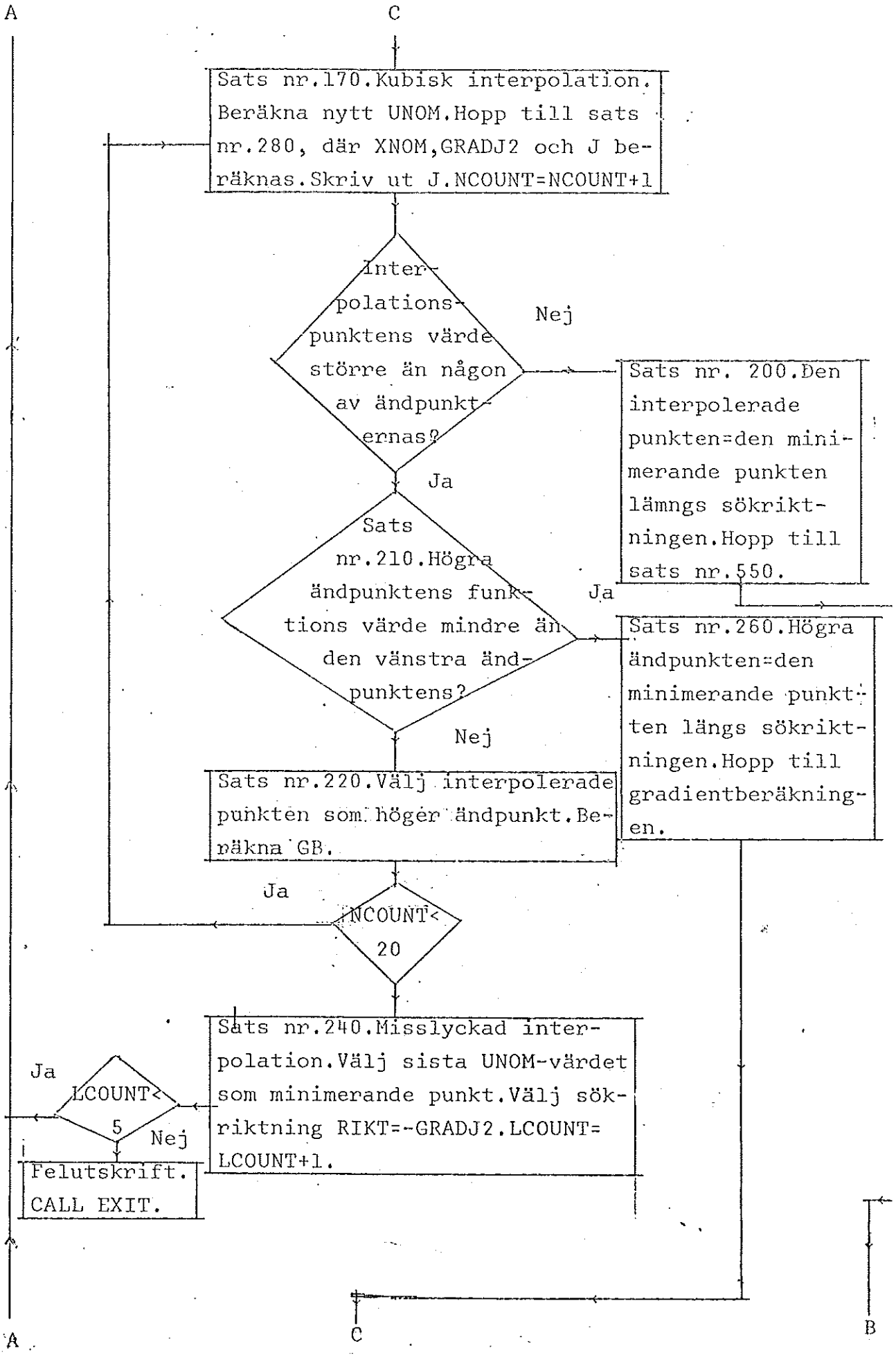
5. Flödesschema för program ICG1.

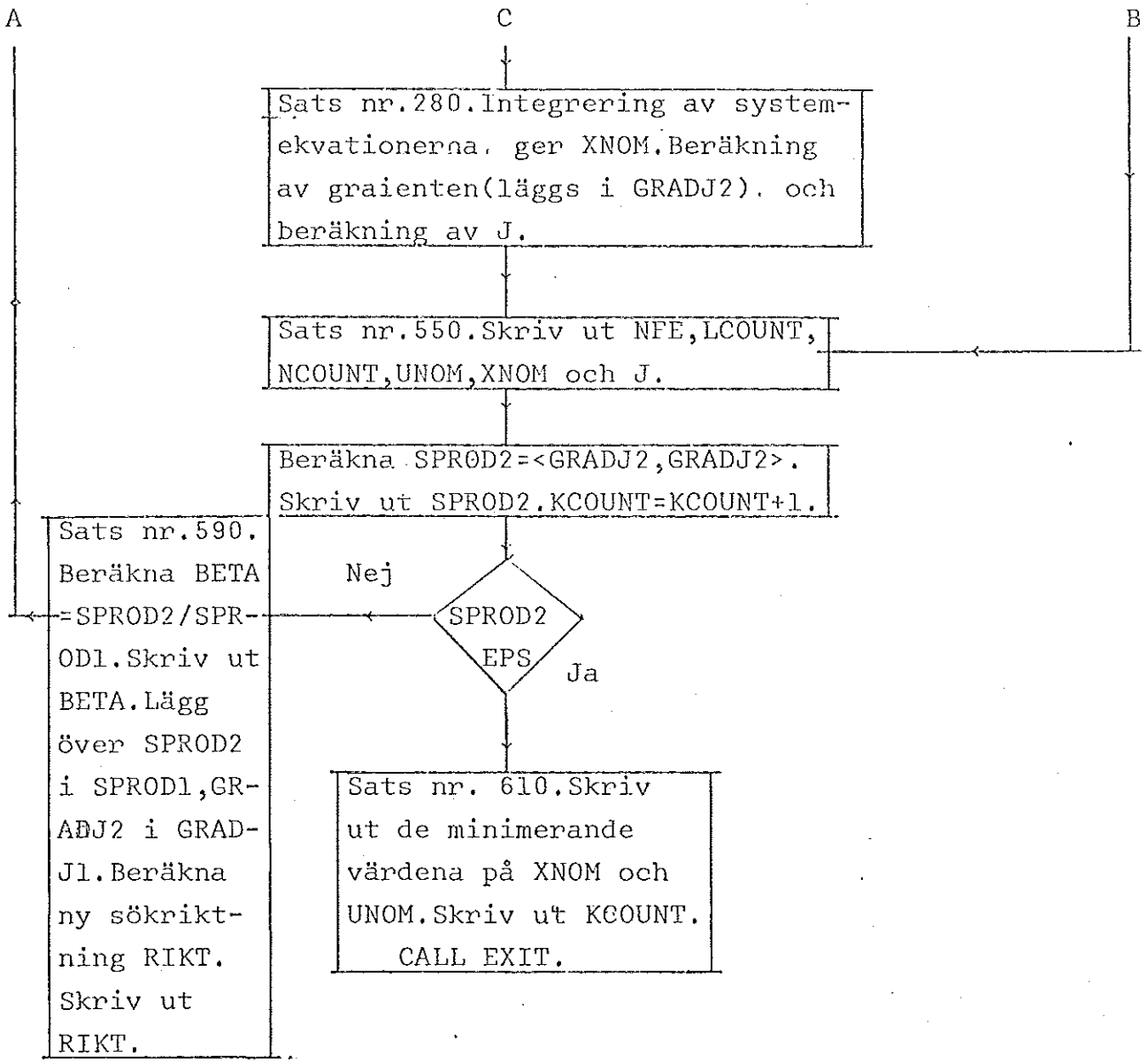
Nedan anges flödesschemat för program ICG1. Detta program baserar sig på den modifierade varianten av Fletcher-Reeves algoritm, som beskrivs i kap.4 och vars flödesschema finns i appendix A.
Flödesschemat visar i stort, hur programmet fungerar, utan att gå in på själva beräkningarna, vilka istället beskrivs i kap.7,8,9 och 10.Däremot betonas mer utskrifter och hopp, varför satsnummer från programmet har satts ut i flödesschemat vid hopp. För förklaring av symboler mm. och förtydligande av själva programmet se kap.6.

Program ICG1.




C
Sats nr.70 Start av minimering
Beräkna riktningsderivatan GB= GRADJ2 RIKT
Nej
GB≤0
Ja
Sats nr.90.Sätt RIKT=-RIKT och GB=-GB.Skriv RIKT=-RIKT.
Sats nr.110.Nollställ MCOUNT (räknare för antalet extra-polationer).
Sats nr.120.Start av extra-polation.MCOUNT=MCOUNT+1.
Beräkna nytt UNOM.Hoppa till sats nr.280,där nya XNOM,GRADJ2 och J beräknas.Skriv ut J.Be-räkna GB.
Inter-polation eller extrapol-ation?
Sats nr.160. Startinter-polation.
Skriv ut MCOUNT.Noll-ställ NCOUNT (räknare för antal inter-polationer).
Sats nr. 155.MCOUNT≤ 20
Nej
Sats nr.1017.Felutskrift.CALL EXIT.
A
C


A
C
Sats nr.170.Kubisk interpolation. Beräkna nytt UNOM.Hopp till sats nr.280, där XNOM,GRADJ2 och J beräknas.Skriv ut J.NCOUNT=NCOUNT+1
Inter-polations-punktens värde större än någon av ändpunkt-ernas?
Nej
Sats nr. 200.Den interpolerade punkten=den minimerande punkten lämngs sökriktningen.Hopp till sats nr.550.
Ja
Sats nr.210.Högra ändpunktens funktions värde mindre än den vänstra ändpunktens?
Ja
Sats nr.260.Högra ändpunkten=den minimerande punkten längs sökriktningen.Hopp till gradientberäkningen.
Nej
Sats nr.220.Välj interpolerade punkten som höger ändpunkt.Beräkna GB.
Ja
NCOUNT< 20
Sats nr.240.Misslyckad interpolation.Välj sista UNOM-värdet som minimerande punkt.Välj sökriktning RIKT=-GRADJ2.LCOUNT=LCOUNT+1.
Ja
LCOUNT< 5
Nej
Felutskrift. CALL EXIT.
A
C
B


A
C
B
Sats nr.280.Integrering av system-
ekvationerna, ger XNOM.Beräkning
av graienten(läggs i GRADJ2). och
beräkning av J.
Sats nr.550.Skriv ut NFE,LCOUNT,
NCOUNT,UNOM,XNOM och J.
Beräkna SPROD2=<GRADJ2,GRADJ2>.
Skriv ut SPROD2.KCOUNT=KCOUNT+1.
Sats nr.590.
Beräkna BETA
=SPROD2/SPR-
OD1.Skriv ut
BETA.Lägg
över SPROD2
i SPROD1,GR-
ADJ2 i GRAD-
J1.Beräkna
ny sökrikt-
ning RIKT.
Skriv ut
RIKT.
Nej
SPROD2
EPS
Ja
Sats nr. 610.Skriv
ut de minimerande
värdena på XNOM och
UNOM.Skriv ut KCOUNT.
CALL EXIT.

6. Förtydligande och förklarande av programmet.

I programmet approximeras tidsfunktionen (t.ex. x(t),u(t)) med värden i ett antal ekvidistanta tidspunkter. Tidsintervallet indelas därför i NSTEP intervall. Varje funktion kommer då att motsvaras av ett fält med ett index, där varje komponent är funktionsvärdet 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 sättes lika med 1 vid starttidpunkten T0($=t_0$) och lika med NSTP=NSTEP+1 vid sluttidpunkten T1($=t_1$). I anger den aktuella tidpunkten t genom sambandet T=T0+(I-1)*H, där H är intervallängden. Nedan ges sambandet mellan olika vektorer och motsvarande fält.

x(t) ↔ XNOM(K,I) K=1,.....,N
u(t) ↔ UNOM(J,I) J=1,.....,NU
Sökriktning d(t) ↔ RIKT(K,I) K=1,.....,NU
Gradienten m.a.p. u=g(t) ↔ GRADJ1 eller
GRADJ2 K=1,.....,NU

Vidare används följande beteckningar:

n ↔ N
m ↔ NU
$x(t_0)$ ↔ XIN(K) K=1,.....,N
$F(x(t_1))$ ↔ SF
f^k ↔ FVEC(K) K=1,.....,N $f = \begin{bmatrix} f^1 \\ \vdots \\ f^n \end{bmatrix}$
f_x^k ↔ FX(K,L) K=1,.....,N L=1,....,N
$\left(\frac{\partial f}{\partial \bar{x}_l} = \text{FX(K,L)}\right)$
f_u^j ↔ FU(K,J) K=1,.....,N J=1,.....,NU
$\left(\frac{\partial f^k}{\partial \bar{u}_j} = \text{FU(K,J)}\right)$

$L \leftrightarrow SL$

$L_x \leftrightarrow SLX(K)$ $\qquad K=1,\ldots\ldots,N$

$\left(\frac{\partial L}{\partial x_k} = SLX(K) \right)$

$L_u \leftrightarrow SLU(J)$ $\qquad J=1,\ldots\ldots,NU$

$\left(\frac{\partial L}{\partial u_j} = SLU(J) \right)$

$t_0 \leftrightarrow T0$

$t_1 \leftrightarrow T1$

$g_{k+1}, g_{k+1} \leftrightarrow$ SPROD1 eller SPRØD2

$\beta \leftrightarrow BETA$

$\varepsilon \leftrightarrow EPS$

Varje gång UNOM ändras i programmet, beräknas i gradientberäkningen ett nytt XNOM, där integration av systemekvationerna ingår.
I programmet används en subrutin QSF, som fungerar på följande sätt:
Subrutinen utför integration av en ekvidistant tabulerad funktion med hjälp av Simpsons regel. Om dessa funktionsvärden y_i är givna vid ekvidistanta punkter $x_i=a+(i-1)\cdot h$, beräknar QSF fältet Z, där $Z_i=\int_a^{x_i} y(x)dx$, $i=1,\ldots,n$. Om funktionsvärdena läggs i fältet Y, intervallängden är H och NDIM är dimensionen av vektorerna Y och Z, fås fältet Z genom anropet QSF(H,Y, Z,NDIM). I detta program har konsekvent använts Q,SINT och NSTP för Y,Z och NDIM. För närmare studium av subrutinen hänvisas till [5].
Beräkning av gradienten av J m.a.p. u och beräkning av funktionsvärdet av J görs på ett ställe i programmet och utnyttjas genom hopp från de ställen i programmet, där dessa värden behöver uträknas. Beroende på vad variabeln IND har för för värde, sker sedan genom väljarstyrt hopp återhopp till rätt ställe i programmet.
Programmets funktion i stort:
Programmet får först initialvärden från subrutinen USER (se närmare kap.8). Som första sökriktning väljes

den negativa gradienten. Minimering längs denna sökriktning sker (se kap.7), varpå programmet testar om skalärprodukten av gradienten med sig själv (SPROD2) har blivit tillräckligt liten. (mindre än EPS). Om så är fallet har minimum nåtts och programmet skriver ut de minimerande x- och u-värdena. I annat fall börjar vi en ny minimering med de gamla x- och u-värdena som startvärden.

Utskrifter:

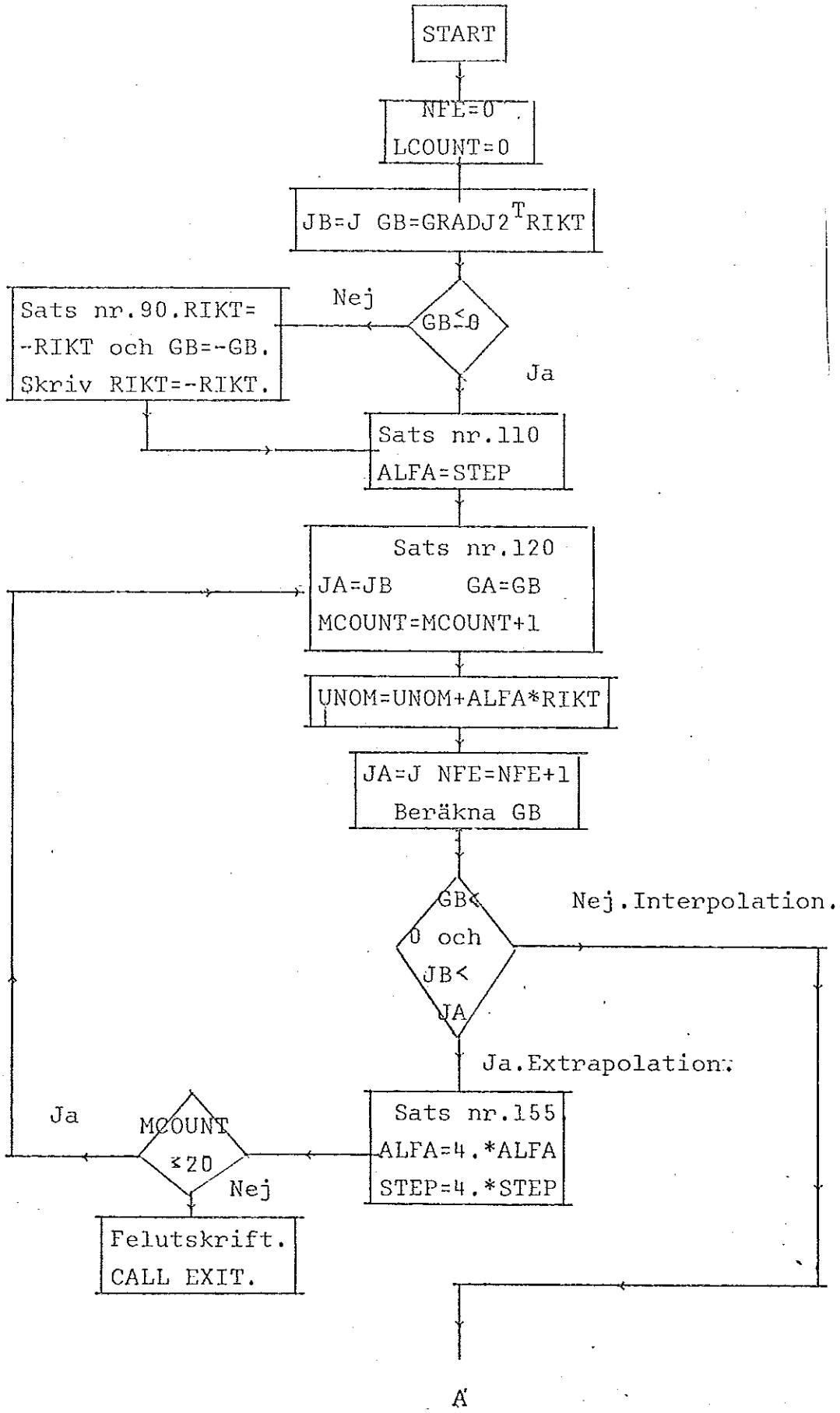
Först skrivs en del initialvärden ut såsom N,NU,T0, T1,H,NSTEP,NPRINT(se kap.8),XIN och de gissade startvärdena på UNOM .Därefter skrivs det från systemekvationerna framintegrerade XNOM ut, samt det då beräknade J-värdet. Gradienten GRADJ2 och dess skalärprodukt med sig själv SPROD1 skrivs också ut. Detta görs bara en enda gång i programmet, nämligen då den första sökriktningen sätts lika med negativa gradienten. Inuti minimeringsslingan för en speciell riktning skrivs först ut, hur stort steg (ALFA) som tages och sedan vilka värden J och riktningsderivatan GB får vid extrapolation, samt antalet extrapolationer (MCOUNT),då interpolationen börjar. Därefter skrivs ut, hur stort interpolationssteg som tages (SLAM), vilka värden J och riktningsderivatan GB får vid interpolationen,samt vilken punkt som tagits som minimerande punkt och antalet interpolationer (NCOUNT). Vidare skrivs också NFE (antalet beräkningar av J i minimeringsslingan), LCOUNT (antalet misslyckade interpolationer), de minimerande värdena på UNOM och XNOM för den aktuella riktningen (se kap.7), samt värdet på SPROD2. Om SPROD2 inte är tillräckligt liten, skrivs BETA=(skalärprodukten av den nya gradienten med sig själv(SPROD2))/(skalärprodukten av den gamla gradienten med sig själv (SPROD1)) och den nya sökriktningen RIKT ut.STEP sättes lika med STEG, som är det ursprungliga värdet på STEP (det som finns i USER).

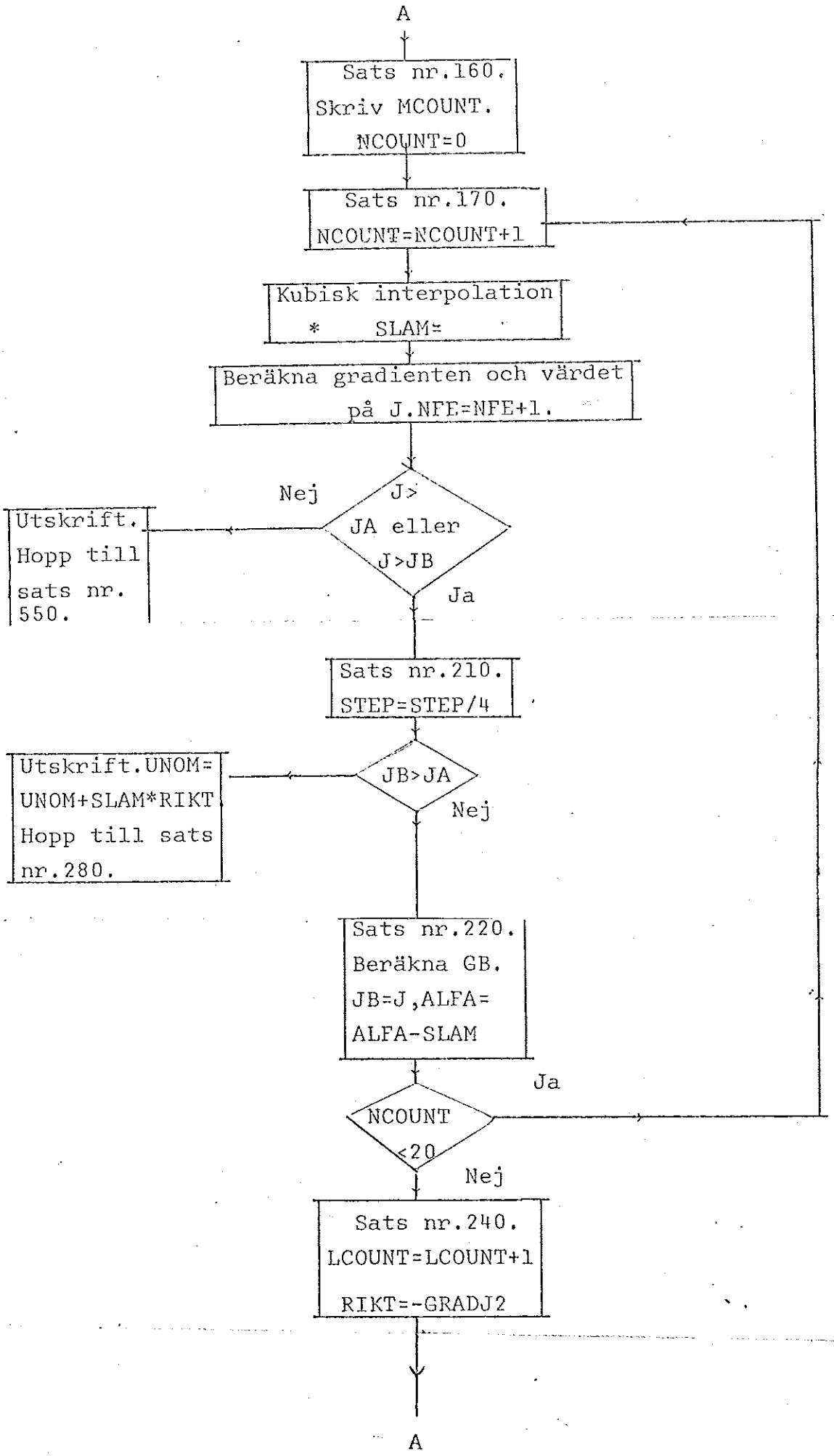
Om SPROD2 är tillräckligt liten, skrivs antalet riktning-

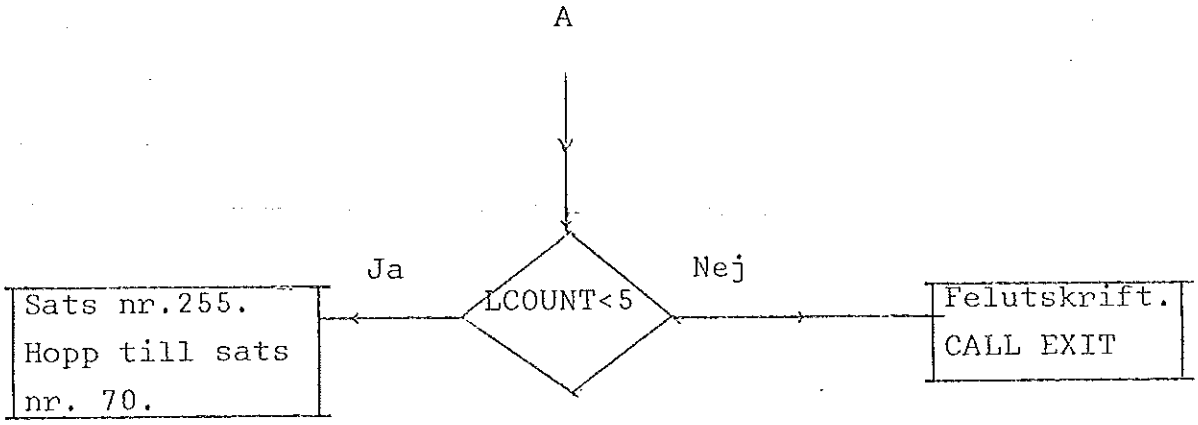
ar (KCOUNT), som programmet har minimerat längs, ut. Dessutom skrivs de optimala värdena på XNOM och UNOM ut.

7. Beskrivning av den endimensionella minimeringen.

Flödesschema




A
Sats nr.160.
Skriv MCOUNT.
NCOUNT=0
Sats nr.170.
NCOUNT=NCOUNT+1
Kubisk interpolation
* SLAM=
Beräkna gradienten och värdet
på J.NFE=NFE+1.
J>
JA eller
J>JB
Nej
Utskrift.
Hopp till
sats nr.
550.
Ja
Sats nr.210.
STEP=STEP/4
JB>JA
Utskrift.UNOM=
UNOM+SLAM*RIKT
Hopp till sats
nr.280.
Nej
Sats nr.220.
Beräkna GB.
JB=J,ALFA=
ALFA-SLAM
Ja
NCOUNT
<20
Nej
Sats nr.240.
LCOUNT=LCOUNT+1
RIKT=-GRADJ2
A




```
*   Z=3.*(JA-JB)/ALFA+GA+GB      W=SQRT(Z**2-GA*GB)
SLAM=ALFA*(GB+W-Z)/(GB-GA+2.*W)
```

Satsnummer har hämtats från programmet.

Förklaring av symboler:

NFE= antal funktionsberäkningar.
STEP=steglängd, som används vid nästa minimering.
ALFA= steglängd vid extrapolation.
MCOUNT=antal extrapolationer vid en minimering.
NCOUNT= antal interpolationer vid en minimering.
J= aktuellt funktionsvärde(kallas i programmet SJ, efersom detta tal är reellt).
JA= funktionsvärde i den vänstra ändpunkten.
JB= funktionsvärde i den högra ändpunkten.
(JA och JB är i programmets början deklarerade som REAL).
GB= riktningsderivatan= GRADJ2^T*RIKT.
GRADJ2= gradienten i aktuell punkt.
RIKT= aktuell sökriktning.

Funktion:

Vi minimerar längs en sökriktning RIKT, genom att ändra UNOM längs denna riktning enligt UNOM=UNOM+ALFA*RIKT.

Metoden börjar med att beräkna funktionsvärdet och riktningsderivatan för givet UNOM (punkt A). För att få ett lägre funktionsvärde, när man extrapolerar åt höger, skall riktningsderivatan vara mindre än noll (se fig 1), i annat fall väljes sökriktningen motsatt tidigare sökriktning och "RIKT=-RIKT" skrives ut.

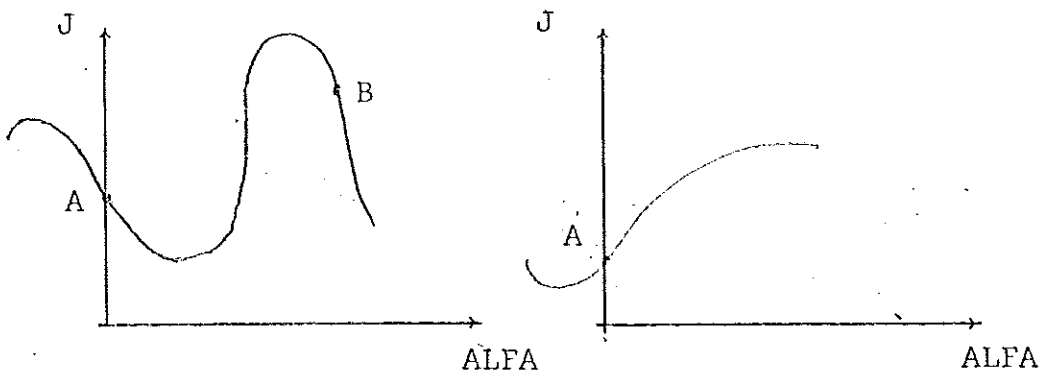


Fig 1a) Negativ riktningsderivata Fig 1b) Positiv riktnings-derivata.

Därefter görs en extapolation med steglängden ALFA och funktionsvärde samt riktningsderivata beräknas i denna punkt (punkt B). Om JA<JB (fig 2a) eller GB>0 (fig 2b), så startas interpolationen, i annat fall multipliceras med fyra och ny extrapolation sker.

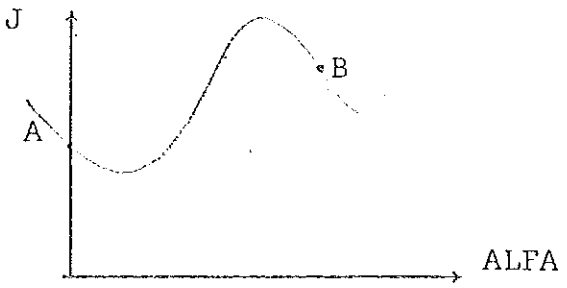


Fig 2a) JA<JB

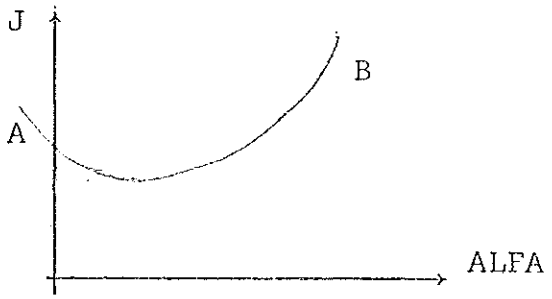


Fig. 2b) GB>0

Om 20 interpolationer gjorts i rad, avbryts programmet och "EXTRAPOLATION 20 TIMES.END OF PROGRAM." skrivs ut.

Vid interpolationens början skrivs ut antalet gjorda extrapolationer ut "MCOUNT=".Saedan görs en kubisk interpolation, varvid man anpassar ett tredjegrads-polynom till kurvan. Därvid beräknas hjälpstorheterna Z,W och SLAM, som anger hur långt steg tillbaka, som man skall ta från den högra punkten vid interpolation (se fig 3).

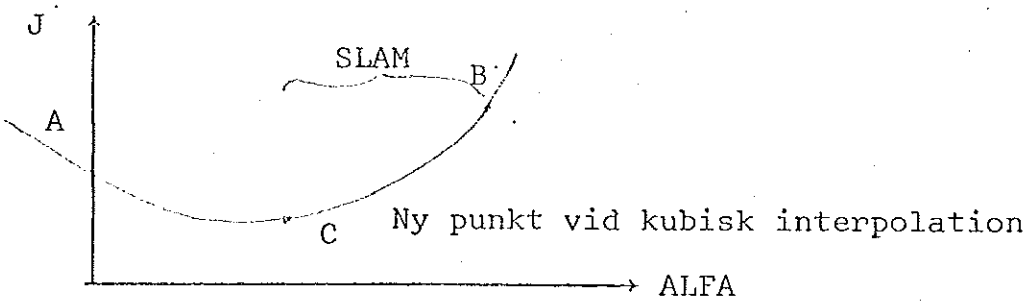


Fig 3 Kubisk interpolation.

I den nya punkten (punkt C) beräknas funktionsvärdet, och om detta är mindre än ändpunkternas funktions-värden JA och JB, så är minimeringen färdig (se fig 4a) och "THE MINIMIZING POINT ALONG THE SEARCHDIRECTION= THE INTERPOLATED POINT" skrivs ut. Om så inte är fallet divideras steglängden med fyra, och om JB<JA väljes

B som minimerande punkt och "THE MINIMIZING POINT= THE RIGHT ENDPOINT" skrivs ut (se fig 4b).Om JA<JB och JA<JC väljes den interpolerade punkten som höger ändpunkt och ny interpolation görs (se fig 4c).

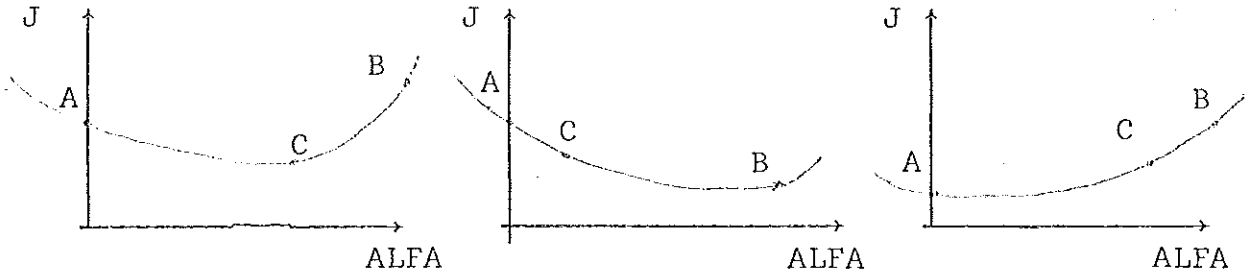


Fig 4a)JC<JA och JC≤JB.

Fig 4b) JB<JA och JC>JB.

Fig 4c) JA<JB och JA<JC.

Om 20 interpolationer görs i sträck avbryts minimeringen och omstart sker av Fletcher-Reeves med sökriktning lika med negativa gradienten i den aktuella punkten."INTERPOLATION FAILED NCOUNT=20" skrivs ut.

Kommentar: Avsikten med minimeringen är, att finna en punkt i närheten av ett minimum längs sökriktningen. För att minimeringen skall fungera tillfredsställande bör funktionen ej vara alltför komplicerad. Vid den kubiska interpolationen kan uttrycket Z^2-GA*GB , vilket står under rottecken, bli negativt, varför detta uttryck i så fall sättes lika med noll.
Efter minimeringen skriver programmet ut värden på NFE,NCOUNT,LCOUNT,XNOM,UNOM och J. LCOUNT är en räknare, som anger antalet omstarter, då mer än 20 interpolationer gjorts i rad. Om 5 sådana omstarter har gjorts avbryts programmet och "LCOUNT=5" skrivs ut.

8. Beskrivning av subrutinen USER.

För att göra programmet generellt användes en subrutin USER, som innehåller alla data, som behövs för ett specifikt problem.För att slippa anropa subrutinen med parametrar varje gång har subrutinen och huvudprogrammet en gemensam minnesarea, ett COMMON-block där alla aktuella värden finns.

Subrutinen USER används, dels till att ge initialvärden till huvudprogrammet, dels till att vid en bestämd tidpunkt räkna ut ett antal funktionsvärden t.ex. partiella derivator,SF,SL och F. När USERberäknar dessa funktionsvärden hämtar den de aktuella värdena på x och u i COMMON-blocket. Innan anropet av USER har nämligen i huvudprogrammet värdena på XNOM och UNOM vid den aktuella tidpunkten överlagrats i arbetsvektorerna XVEC resp. UVEC, vilka har samma minnesplats i COMMON-blocket som de fält x och u, vilka används i USER.

Subrutinen USER bstår av ett antal underavdelningar med olika ENRTY POINTS, vilka kan anropas var för sig med en CALL-instruktion.Vi kommer nedan beskriva de olika avdelningarnas användning.

ENTRY INIT: I denna avdelning ges initialvärdena till huvudprogrammet. N anger antalet tillståndsvariabler och NU antalet styrvariabler. NSTEP är antalet intervall, som det aktuella tidsintervallet från T0 (starttidpunkt)till T1 (sluttidpunkt) har indelats i, varvid intervallängden blir H=(T1-T0)/NSTEP. Obs! Glöm ej att ändra H och NPRINT vid ändring av NSTEP. NPRINT anger hur många punkter man vill skriva ut. Om t.ex. NPRINT sättes lika med 10 skrivs var tionde punkt i tidsintervallet ut. Randvärden för XNOM finns i XIN. Begynnelsvärdet på UNOM, dvs. initialgissningen, placeras också här. EPS anger hur liten skalärprodukten med sig själv skall vara, för att hela programmet skall vara färdigt. STEP är steglängden vid extrapolation första gången vi minimerar och vid omstart (NCOUNT>20). Sedan modifierar

programmet självt STEP i minimeringen.

ENTRY FINLOS:SF är den funktion i J, som endast beror på XNOM:s sluttillstånd.

ENTRY INTLOS: SL är den funktion i J, som finns under integraltecknet.

ENTY BBOUND: Här finns de partiella derivatorna av SF m.a.p. x.

ENTRY PDERFH: Här står de partiella derivatorna av F och SL m.a.p. x och u.

ENTRY SYSTEM: Här anges systemekvationerna.

Ett exempel på hur en USER ser ut, finns i appendix B.

9. Beskrivning av gradientberäkningen.

Enligt teorin för beräkning av gradienten av J m.a.p. u (se kap 4), består gradientberäkningen av tre delar. I två av dessa delar måste en ekvation integreras, och den integrationsmetod, som då användes, skall jag först beskriva.

Den integrationsmetod, som jag har använt, är en fjärde ordningens Runge-Kutta metod. Utformningen av integrationen har jag hämtat från en subrutin RK1ST(T,YIN, H,YE,N,IA),som integrerar $\frac{dx}{dt}=f(x,t)$ ett steg H.Den ser ut på följande sätt:

```
      SUBROUTINE RK1ST(T,YIN,H,YE,N,IA)
C     T=ACTUAL TIME
C     YIN=ACTUAL VALUE OF X
C     H=INTEGRATION STEP LENGTH
C     YE=VALUE OF X AT TIME T+H
C     N=NUMBER OF EQUATIONS
C     IA=DIMENSION PARAMETER
C
C     THE FUNCTION VALUES F(X,T) ARE
C     PROVIDED VIA A SUBROUTINE FUNC.
C     T,X,DX/DT LIE IN A COMMON BLOCK /FUNCTION/
C
      DIMENSION YIN(IA),YE(IA),W(10),Z(10),A(5)
      COMMON /FUNCTION/ TE,W,Z
      A(1)=A(2)=A(5)=H/2.
      A(3)=A(4)=H
      TE=T
      DO 10 I=1,N
   10 YE(I)=W(I)=YIN(I)
      DO 20 J=1,4
      CALL FUNC
      TE=T+A(J)
      DO 20 K=1,N
      W(K)=YIN(K)+A(J)*Z(K)
   20 YE(K)=YE(K)+A(J+1)*Z(K)/3.
      RETURN
      END
```

Kommentar: I mitt fall ligger T,X,DX/DT i ett COMMON-block /USER/ och funktionsvärdena f(x,t) fås via anrop av USER. W är en arbetsvektor, som behövs vid beräkningarna. A är en vektor, som används, för att vi skall kunna skriva själva integrationen så lätt som möjligt.

Eftersom vi känner λ vid sluttidpunkten ($\lambda(t_1)=F_x^T(\bar{x}(t_1))$), kan gradienten beräknas direkt vid denna tidpunkt utan att integrera $\dot{\lambda}=-L_x^T-f_x^T\lambda$, vilket gör att punkt 2 och 3 byter plats i den sammanfattning av gradientberäkningen, som har gjorts i kap 4.

1. Integrera $\dot{x}=f(x,u)$ $x(t_0)=x_0$ från t_0 till t_1.

Integrationen har gjorts enligt subrutin RK1ST, där T,H,YIN,W,YE,N och IA motsvaras av TE,H,XNOM(J,I), XVEC,XNOM(J,I+1),N och N. Vid starttidpunkten (t_0) ges XNOM värden från XIN i USER p.g.a. att ekvationen $\dot{x}=f(x,u)$ har begynnelsevärdet $x(t_0)=x_0$. XIN har fått värden i början av programmet, där USER anropades för att få fram en del initialvärden. Vid integrationen görs dessutom en linjär interpolation i UNOM, och därvid används vektorerna BM och BP, vila har fått värden i början av programmet. Följande programavsnitt gällande punkt 1 har då skrivits:

```
C
C     INTEGRATE DX/DT=F(X,U,T) TO
C     GET XNOM.RUNGE-KUTTA METHOD.
C
      DO 290 I=1,N        Kommentar: x(t0)=x0
  290 XNOM(I,1)=XIN(I)
      T=T0
      DO 330 I=1,NSTEP
      TE=T
      DO 300 J=1,N
      XVEC(J)=XNOM(J,I)
  300 XNOM(J,I+1)=XVEC(J)
      DO 320 J=1,4
      DO 310 K=1,NU
  310 UVEC(K)=(BP(J)*UNOM(K,I)+BM(J)*UNOM(K,I+1))/2.  Kommentar:Interpolation
      CALL SYSTEM  Kommentar:Beräkning av FVEC med                i UNOM
      TE=T+A(J)                aktuella värden på x,u och t.
      DO 320 K=1,N
      XVEC(K)=XNOM(K,I)+A(J)*FVEC(K)
  320 XNOM(K,I+1)=XNOM(K,I+1)+A(J+1)*FVEC(K)/3.
      T=T+H
  330 CONTINUE
```

Mellan integreringen av $\dot{x}=f(x,u,t)$ och gradientberäkningen finns ett avsnitt, som bara används vid sluttidpunkten (t_1) och där randvärdet till ekvationen $\dot{\lambda}=-L_x^T-f_x^T\lambda$ beräknas ($\lambda(t_1)=F_x(\bar{x}(t_1))$). Nedanstående programavsnitt har då skrivits:

```
C
C     COMPUTE THE GRADIENT AND PUT IT IN GRADJ2.
C
C
C     PUT LAMBDA(T1) IN SLIN.
C
      TE=T1
      DO 335 J=1,N
  335 XVEC(J)=XNOM(J,NSTP)
      CALL BBOUND
      DO 340 J=1,N
  340 SLIN(J)=SFX(J)
      T=T1
```

Kommentar: SFX får värde genom anrop av ENTRY BBOUND i vid sluttidpunkten

Se kommentar på gradientberäkningen.

Efter integrering av systemekvationerna kommer en slinga, där gradienten beräknas och ekvationen $\dot{\lambda}=-L_x^T-f_x^T$ integreras baklänges. GCOUNT användes som räknare i denna slinga.

2 Beräkna $\nabla_u J(\bar{u},t)=L_u^T(\bar{x}(t),\bar{u}(t))+f_u^T(\bar{x}(t),\bar{u}(t))\lambda(t)$ och lägg den i GRADJ2.

L_u är en radvektor med m (NU) komponenter, medan f_u är en nxm (NxNU) matris. $\nabla_u J$ är en kolonnvektor med m (NU) komponenter. Slutligen är λ en radvektor med n (N) komponenter. För en bestämd tidpunkt blir

$$\nabla_u J(t)=\begin{bmatrix} L_{u_1}+\sum_{k=1}^{n} f_{u_1^*}^{i}\,\lambda_i \\ \vdots \\ L_{u_m}+\sum_{k=1}^{n} f_{u_m^*}^{i}\,\lambda_i \end{bmatrix}$$

Följande programavsnitt har då skrivits:

```
C
C     LOOP WHERE THE GRADIENT=LUT+
C     FUTLAMBDA IS COMPUTED AND
C     LAMBDA BACKWARDS INTEGRATED.
C     XNOM AND UNOM ARE INTERPOLATED.GCOUNT IS COUNTER IN THAT LOOP.
C
C
C     COMPUTE THE GRADIENT AND
C     PUT IT IN GRADJ2.
C
      GCOUNT=0
  350 DO 360 J=1,N
      SLVEC(J)=SLIN(J)
  360 XVEC(J)=XNOM(J,NSTP-GCOUNT)
      DO 370 K=1,NU
  370 UVEC(K)=UNOM(K,NSTP-GCOUNT)
      TE=T
      CALL PDERFH
```

Kommentar: Vid sluttidpunkten måste vektorerna SLIN och SLVEC, som används vid integrering av $\dot{\lambda}=-L_x^T-f_x^T$,få värden(de sättes lika med SFX, dvs. det är $\lambda(t_1)=F_x^T(x(t_1))$som uppfylls).

```
      DO 420 J=1,NU
      Q(J)=0.
      DO 410 K=1,N
  410 Q(J)=Q(J)+SLVEC(K)*FU(K,J)
  420 GRADJ2(J,NSTP-GCOUNT)=SLU(J)+Q(J)
C
C     TEST END OF LOOP.
C
      IF(GCOUNT.EQ.NSTEP) GO TO 500
```

Lägg märke till, att som tidsindex används NSTP-GCOUNT, och då GCOUNT ökar med 1 varje gång vi går igenom slingan, minskar tidsindexet, dvs. vi går baklänges i tiden.Jag testar här, om vi kommit tillbaka till starttidpunkten, eftersom först integreras $\dot{\lambda}=-L_x^T-f_x^T$ (vilket görs efter gradientberäkningen i programmet) och sedan hoppar det upp och beräknar gradienten.

3. Integrera $\dot{\lambda}=-L_x^T-f_x^T$ $\lambda(t_1)=F_x(\bar{x}(t_1))$ från t_1 till t_0.

L_x är en radvektor med n (N) komponenter, medan f_x är en nxn (NxN) matris. Vid en bestämd tidpunkt skall följande integreras:

$$\dot{\lambda}(t) = \begin{bmatrix} -L_{x_1} - \sum_{k=1}^{n} f_{x_1}^{i} * \lambda_i \\ \vdots \\ -L_{x_n} - \sum_{k=1}^{n} f_{x_n}^{i} * \lambda_i \end{bmatrix}$$

Integrationen har gjorts enligt subrutin RK1ST, där T,H,YIN,W,YE,N och IA motsvaras av TE,H,SLIN,SLVEC, SLUT,N och N. Vid integrationen görs interpolationer (linjära) i XNOM och UNOM på samma sätt som då system-ekvationerna löstes. Följande programavsnitt har då erhållits:

```
C
C     INTEGRATION OF -DLAMDA/DT=LXT+FXTLAMBDA
C     WITH RUNGE-KUTTA METHOD.INTERPOLATION OF XNOM AND UNOM.
C
      DO 430 J=1,N
  430 SLUT(J)=SLIN(J)
      DO 480 J=1,4
      DO 440 K=1,N
  440 XVEC(K)=(BP(J)*XNOM(K,NSTP-GCOUNT-1)+BM(J)*XNOM(K,NSTP-GCOUNT))/2.
      DO 450 L=1,NU
  450 UVEC(L)=(BP(J)*UNOM(L,NSTP-GCOUNT-1)+BM(J)*UNOM(L,NSTP-GCOUNT))/2.
      TE=T-A(J)                    Kommentar: Interpolation i XNOM och UNOM
      CALL PDERFH
      DO 470 L=1,N
      Q(L)=0.
      DO 460 K=1,N
  460 Q(L)=Q(L)+SLVEC(K)*FX(K,L)
  470 SH(L)=SLX(L)+Q(L)            Kommentar:SH=L_x^T+f_x^T λ
      DO 480 K=1,N
      SLVEC(K)=SLIN(K)+A(J)*SH(K)
  480 SLUT(K)=SLUT(K)+A(J+1)*SH(K)/3.
      DO 490 K=1,N
  490 SLIN(K)=SLUT(K)
      T=T-H
      GCOUNT=GCOUNT+1
      GO TO 350
```

10. Beskrivning av beräkning av J.

$$J(u)=F(x(t_1))+\int_{t_o}^{t_1} L(x,u)\,dt$$

Det som står under integraltecknet beräknas vid alla tidpunkter och läggs i Q, varefter QSF anropas och resultatet läggs i SINT(NSTP). Därefter får SF sitt värde genom anrop av ENTRY FINLOS i USER vid slut-tidpunkten och slutsummering sker.

```
C
C     COMPUTE THE LOSS FUNCTION J.
C
      DO 530 I=1,NSTP
      TE=T0+(I-1)*H
      DO 510 L=1,N
  510 XVEC(L)=XNOM(L,I)
      DO 520 K=1,NU
  520 UVEC(K)=UNOM(K,I)
      CALL INTLOS
  530 Q(I)=SL
      CALL QSF(H,Q,SINT,NSTP)
      DO 540 J=1,N
  540 XVEC(J)=XNOM(J,NSTP)
      TE=T0+NSTEP*H
      CALL FINLOS
      SJ=SINT(NSTP)+SF
```

11. Testexempel.

Testexempel 1.

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

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

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

Optimal lösning: $\hat{u}_1(t)=6t-7$

$\hat{x}_1(t)=t^3-\frac{7t^2}{2}+2t+2$

$\hat{x}_2(t)=3t^2-7t+2$

$J_{min}=26$

Vid körning på datamaskin erhölls efter en iteration med NSTEP=500 $J_{min}=26.00011$ (SPROD2=0.00000).

Testexempel 2.

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

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

Förlustfunktion: $J=x_1^2(1)+x_2^2(1)+\int_0^1\{4x_1^2+5x_2^2+u_1^2+u_2^2\}dt$

Optimal lösning: $\hat{u}_1(t)=-e^{-3t}(2-t)$

$\hat{u}_2(t)=-e^{-3t}(1-t)$

$\hat{x}_1(t)=e^{-3t}(1-t)$

$\hat{x}_2(t)=e^{-3t}$

$J_{min}=2$

Vid körning på dator erhölls efter två iterationer, då NSTEP=100, $J_{min}=2.00361$ (SPROD2=0.02185)

Testexempel 3.

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

$\frac{dx_2}{dt}=x_1^2+2x_1e^t$ $\qquad x_2(0)=0$

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

Optomal lösning: $\hat{u}_1(t)=\frac{1}{2}e^t\cdot(t-1)$

$\hat{x}_1(t)=\frac{1}{2}e^t\cdot(\frac{1}{2}t-1)$

$\hat{x}_2(t)=e^{2t}(\frac{1}{8}t^2-\frac{1}{8}t-\frac{7}{16})$

$J_{min}=-2.6459$

För att visa hur den numeriska noggrannheten beror på NSTEP (=antalet intervall som hela tidsintervallet indelas i) anges nedan det J_{min}, som programmet räknar ut vid olika värden på NSTEP.

NSTEP	J_{min}	
20	-2.64562	
100	-2.64587	Jämför teoretiskt värde $J_{min}=-2.64590$
500	-2.64586	

Dessa värden har tagits efter två iterationer (iteration= minimering längs en sökriktning) och värdet på SPROD2 var då 0.00002.Ovanstående siffror visar,att man bör använda ett stort NSTEP för att få god noggrannhet. Man bör dock tänka på,att räknearbetet och därmed exekveringstiden för datorn ökar proportionellt mot NSTEP.

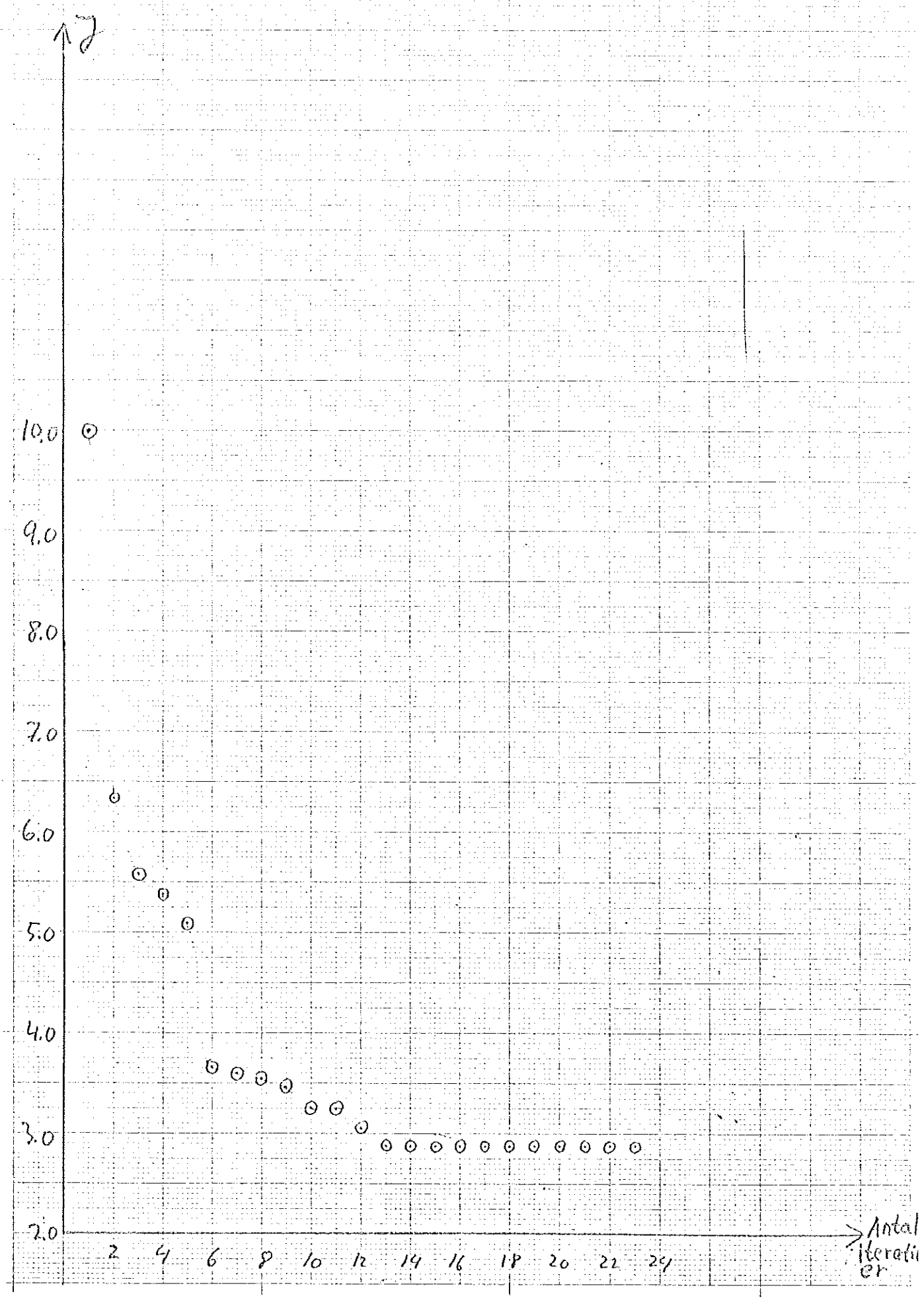
12. Jämförelseexempel.

System: $\frac{dx_1}{dt}=(1-x_2^2)x_1-x_2+u$

$\frac{dx_2}{dt}=x_1$

Som exempel på, hur subrutinen USER används, ges i appendix B den USER, som används för detta exempel. För att kunna jämföra metoden med andra metoder har i diagram 1 avsatts J som funktion av antalet iterationer. Vid körning på dator erhölls efter 23 iterationer, $J_{min}=2.86699$ med NSTEP=500 (SPROD2=0.00004). För studium av hur två andra metoder fungerar på detta exempel, hänvisas till [1] och [2] . Se även kap.15, där mer direkta jämförelser görs mellan min egen metod och de tidigare refererade metoderna.

J som funktion av antal iterationer.



13. Tillämpningsexempel.

Problemet härstammar från en containerterminal. När ett skepp lastar av, förs först containern 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 hjälp av kranarna, kan lossningstiden för båten minskas, vilket är ekonomiskt fördelaktigt.

Då de två kranarna är ungefär lika, behandlas endast lagerkranen här. I den modell som används här, 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 särskilda överföring, som minimering skett för, visas i fig 7. När lastbilen anländer, antas traversen vara belägen, så att den slutliga överföringen kan göras av vagn och vinsch. Problemet reduceras då till 2 dimensioner. Det antas, att avståndet mellan last och vagn ursprungligen är 12 meter, att containern kan anses som en punktmassa och att masscentrum är lika med geometriskt centrum. Alla avstånd refererar till denna punkt. När överföringen börjar, befinner sig vagnen rakt över lastbilen och har ingen hastighet i någon riktning. Dessutom är den vertikala hastigheten hos lasten lika med noll. Vid slutet av överföringen antas på samma sätt systemet vara i vila.

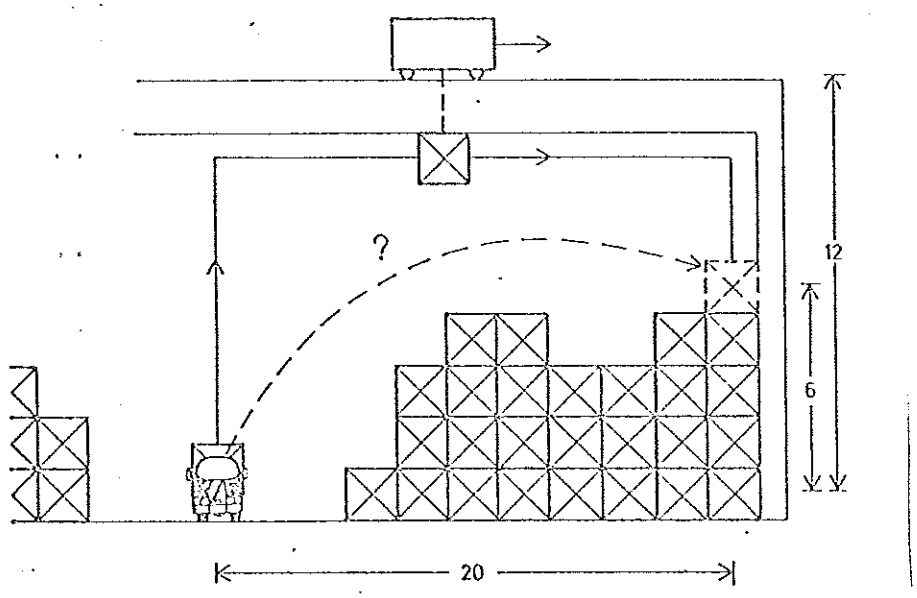
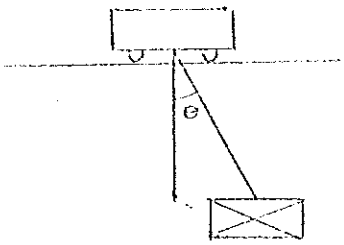


Fig. 1. Den speciella överföringen som minimeras. Alla avstånd i meter.

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

(x_1 = trallans läge, x_1 = trallans hastighet, $x_3 = \Theta$, $x_4 = \dot{\Theta}$, x_5 = linans längd, x_6 = vinsch-hastighet, u_1 = trallans acceleration och u_2 = vinschens acceleration).



$$\begin{aligned}
\dot{x}_1 &= x_2 & x_1(0) &= 0\\
\dot{x}_2 &= u_1 & x_2(0) &= 0\\
\dot{x}_3 &= x_4 & x_3(0) &= 0\\
\dot{x}_4 &= -\frac{g\sin(x_3)-2x_4x_6-u\cos(x_3)}{x_5} & x_4(0) &= 0\\
\dot{x}_5 &= x_6 & x_5(0) &= 12\\
\dot{x}_6 &= u_2 & x_6(0) &= 0
\end{aligned}$$

Under förutsättning att t_1 är känd (t_1 = 20 sek), önskat sluttillstånd enl fig 1 och minimering av den energi, som behövs för operationen erhålls följande förlustfunktion:

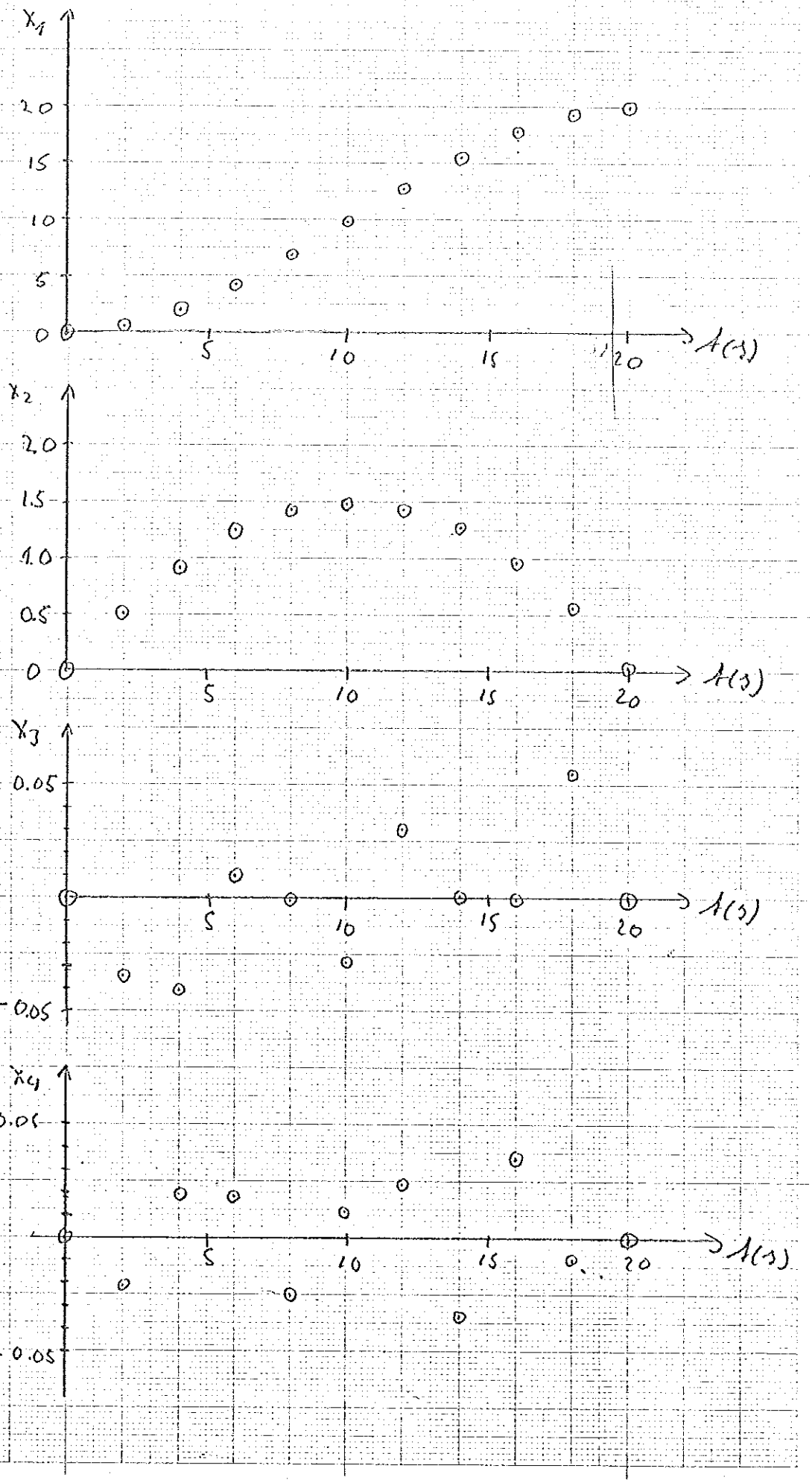
$$J = 10\left((x_1-12)^2 + x_2^2 + x_3^2 + x_4^2 + (x_5-6)^2 + x_6^2\right) +$$

$$+ \int_0^{20} (u_1^2 + u_2^2)\, dt$$

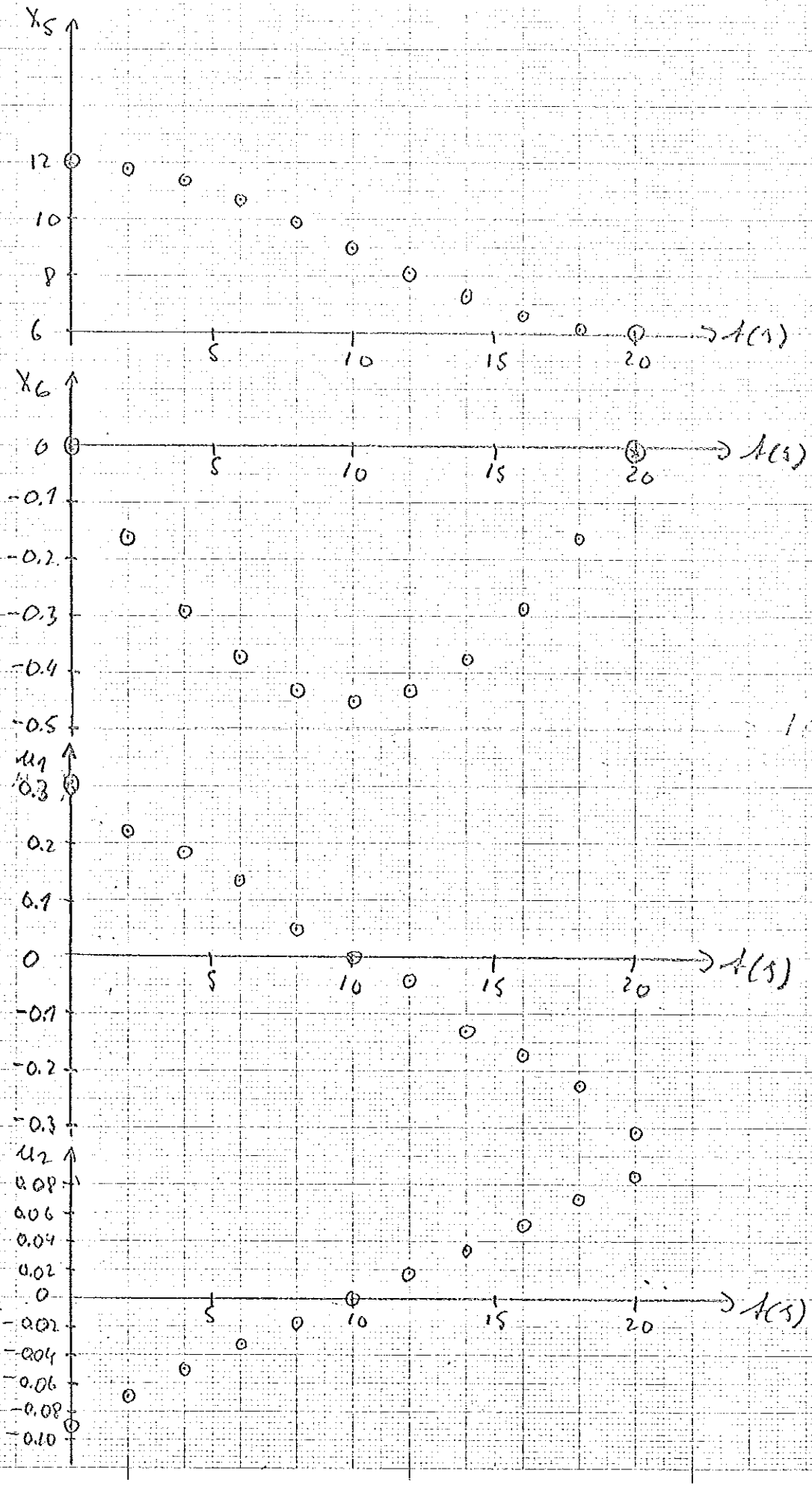
Förutsättningen om sluttillståndet har tagits om hand genom den första delen av J, då avvikelse från sluttillståndet skaffas ganska hårt. Då $u_1^2 + u_2^2$ är proportionell mot den energi som åtgår har $u_1^2 + u_2^2$ valts som L.

Vid körning på dator erhölls de styrstrategier, som visas i diagram 2 och 3 (för NSTEP = 100). Med NSTEP = 500 blev exekveringstiderna mycket långa, varför NSTEP istället valdes till 100. Vidare fick jag första gången räkna ut vilka startvärden jag borde ha på min styrvariabler. Därefter kunde jag som startvärden ha de värden jag fick som slutvärden gången innan. Det minimala värdet på J blev vid 100 iterationer J_{min} = 0.64755 (SPROD2 = 0.00917).

De optimala x_1, x_2, x_3 och x_4 som funktion av tiden.



De optimala x_5, x_6, u_1 och u_2 som funktion av tiden.



14. Egna erfarenheter och diskussion om eventuella förbättringar.

Riktvärden för valda parametrar.

NSTEP: Väljes till 500 så länge problemet inte är omfattande och kräver mycket räknearbete. I så fall får NSTEP minskas (lämpligen till 100), vilket sker på bekostnad av räknenoggrannheten.

Initialgissning av UNOM:Sättes lika med noll vid enklare problem. I annat fall bör rimliga startvärden beräknas. Man kan ta startvärden som approximerar de sista värdena från en tidigare körning, då denna körning ej var tillräcklig.

EPS: Väljes beroende på hur noggrant man vill minimera. Vanliga värden är mellan 0.001 och 0.00001 .

STEP: Vanliga värden är mellan 0.1 och 0.001 .Då steglängden modifieras i den endimensionella minimeringsslingan, har värdet på STEP inte alltför stor betydelse.

Eventuella förbättringar: Det som närmast kan tänkas att förbättras är den endimensionella minimeringen, som bör kunna göras effektivare. Man kan även använda mer sofistikerade integrationsmetoder än den fjärde ordningens Runge-Kutta metod, som jag har använt mig av vid beräkning av gradienten.

15. Jämförelser med program [1] och [2] .

Vi vill först och främst påpeka, att det är väldigt svårt att göra några direkta jämförelser mellan metoderna. Detta beror dels på, att metoderna arbetar på olika sätt, dels p g a att man genom att ändra på vissa parametrar kan konvergenshastigheten ändras betydligt, (gäller program [1] och [2]). Det är svårt att hitta värden på parametrar, så att optimal konvergenshastighet erhålles och dessutom gäller dessa bara för ett specifikt problem. Vi tar upp metod för metod och diskuterar dess för-och nackdelar och jämför den med de andra problemen.

Metod [1] : Se diagram 1 i ref [1] . Samma typ av konvergens som för metod [2] . Konvergenshastigheten verkar vara ungefär som för metod [2] , men sämre än för min egen metod. En nackdel är, att det är svårt att hitta de bästa värdena på EPS1. En fördel är, att problemet kan lösas, utan att systemekvationerna löses och metoden kan enkelt generaliseras till mera komplexa problem, t ex system med tidsfördröjningar.

Metod [2] : Se diagram 1 i ref [2] . En nackdel är svårigheten att välja EPS1 så att onödigt räknearbete göres för varje λ (resp C för metod [1] . En nackdel är att hitta en $C \leq C_0$ så att minimum kan erhållas. Om C är för stort blir det mycket räknearbete vid varje uppdatering av SLAMDA. En fördel, är att systemekvationerna ej löses och metoden kan enkelt generaliseras till mera komplexa problem, t ex system med tidsfördröjningar.

Min egen metod: Se diagram 1. Systemekvationerna måste gå att lösa, vilket är en nackdel. Konvergenshastigheten har varit bra på samtliga testexempel. Eftersom itereringarna sker endast på u, medför det att minimeringen blir enklare.

Dessa jämförelser gäller det jämförelseexempel, som beskrivits i kap 12, men stämmer bra överens för övriga exempel. Överföring av metoderna på program för dator har tagit ungefär

lika lång tid. Programmen har blivit ungefär lika långa, och eftersom de utnyttjar samma huvudprincip, är de till stora avsnitt lika.

16. Referenser.

1 Persson,L,"Körjugerade gradientmetoden för optimala styrproblem",Examensarbete på Inst. för Reglerteknik,Lunds Tekniska Högskola.

2 Johansson,H,"Numerisk lösning av optimala styrproblem via generaliserade multiplikatorfunktioner",Examensarbete på Inst. för Reglerteknik, Lunds Tekniska Högskola.

3 Balakrishan,A.V.,"On a new computing method in optimal control" SIAM Journal on Control,vol.6,1968,pp.149-173.

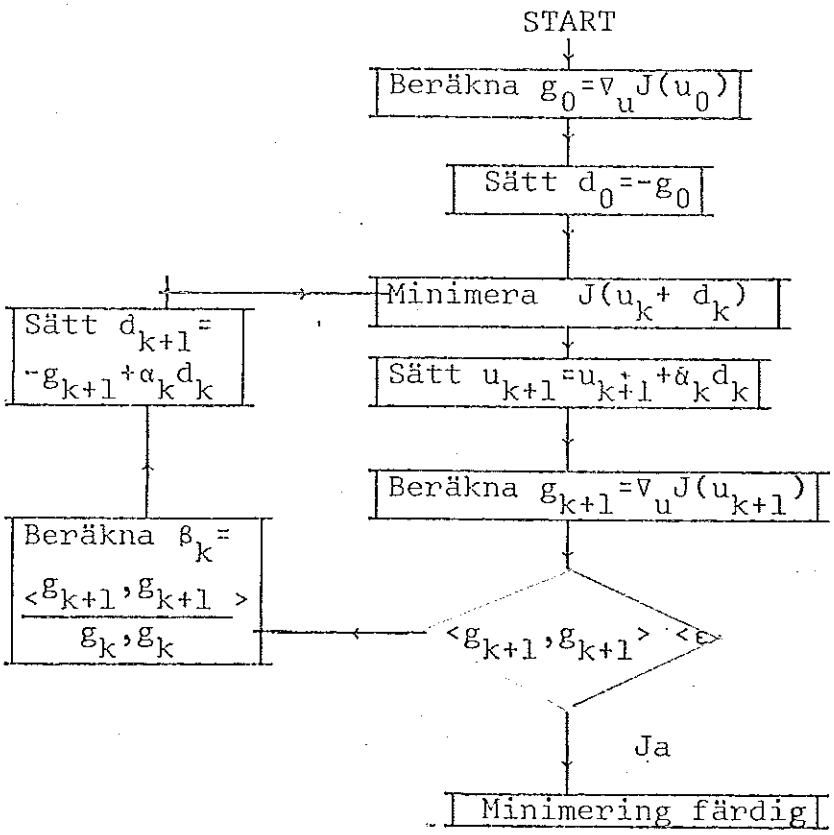
4 Luenberger,D.G.,"Introdution to linear and nonlinear programming" Addisson Wesley,1973.

5 IBM,SSP,Algorithm QSF.

6 Mårtensson,K,"New approaches to the numerical solution of optimal control problems",Studentlitteratur,1972.

Appendix A.

Flödesschema för modifierad Fletcher-Reeves för styrproblemet vid givet u_0.



Appendix B (exempel på en USER).

Den USER, som nedan finns skriven, är den som finns i jämförelseexemplet (se kap.12).

```
      SUBROUTINE USER
C
      DIMENSION XIN(6),UNOM(2,501),X(6),U(2),F(6),FX(6,6),FU(6,2),SFX(6)
     1,SLX(6),SLU(2)
      COMMON/USER/ N,NU,NSTEP,H,T0,T1,EPS,NPRINT,XIN,UNOM,F,SF,SL,SFX,FX
     1,FU,SLX,SLU,T,X,U,STEP
C
      ENTRY INIT
      N=2
      NU=1
      NSTEP=500
      H=0.01
      T0=0.
      T1=5.0
      NPRINT=100
      XIN(1)=0.
      XIN(2)=1.0
      UNOM(1,1)=1.0
      DO 1 I=1,NSTEP
    1 UNOM(1,I+1)=1.0
      EPS=0.0001
      STEP=0.1
      RETURN
C
      ENTRY FINLOS
      SF=0.
      RETURN
C
```

```
      ENTRY INTLOS
      SL=X(1)**2+X(2)**2+U(1)**2
      RETURN
C
      ENTRY BBOUND
      SFX(1)=0.
      SFX(2)=0.
      RETURN
```

```
C
      ENTRY PDERFH
      FX(1,1)=1.0-X(2)**2
      FX(1,2)=-2.*X(2)*X(1)-1.0
      FX(2,1)=1.0
      FX(2,2)=0.
      FU(1,1)=1.0
      FU(2,1)=0.
      SLX(1)=2.*X(1)
      SLX(2)=2.*X(2)
      SLU(1)=2.*U(1)
      RETURN
C
      ENTRY SYSTEM
      F(1)=(1.-X(2)**2)*X(1)-X(2)+U(1)
      F(2)=X(1)
      RETURN
      END
```

Appendix C (listning av program ICG1).

```
C
C     PROGRAM ICG1
C
      DIMENSION UNOM(2,501),XNOM(6,501),FX(6,6),FU(6,2),GRADJ1(6,501),GR
     1ADJ2(8,501),RIKT(8,501),XIN(6),FVEC(6),SLX(6),SLU(2),XVEC(6),UVEC(
     12),Q(501),SINT(501),SFX(6),SLUT(6),SLVEC(6),SLIN(6),A(5),BM(4),BP(
     14),SH(6)
      INTEGER GCOUNT
      REAL JA,JB
      COMMON/USER/ N,NU,NSTEP,H,T0,T1,EPS,NPRINT,XIN,UNOM,FVEC,SF,SL,SFX
     1,FX,FU,SLX,SLU,TE,XVEC,UVEC,STEP
C     INITIALIZE
      CALL INIT
      STEG=STEP
      PRINT 1000
 1000 FORMAT(28H1PRINTOUTS FROM PROGRAM ICG1,//)
      PRINT 1001
 1001 FORMAT(20H AUTHOR LEIF PERSSON,//)
      PRINT 1002,N,NU
 1002 FORMAT(18H NUMBER OF STATES=,I2,5X,28HNUMBER OF CONTROL VARIABLES=
     1,I2,/)
      PRINT 1003,T0,T1
 1003 FORMAT(14H INITIAL TIME=,F10.5,5X,12H FINAL TIME=,F10.5,/)
      PRINT 1004,NSTEP,H
 1004 FORMAT(40H THE TIME INTERVAL T0-T1 IS DIVIDED INTO,I5,36H INTERVAL
     1S (INTEGRATION STEP LENGTH=,F8.5,1H),/)
      PRINT 1005,EPS
 1005 FORMAT(5H EPS=,F8.5,/)
      PRINT 1006,NPRINT
 1006 FORMAT(6H EVERY,I5,55HTH POINT OF THE TRAJECTORIES AND THE CONTROL
     1 IS PRINTED,/)
      PRINT 1007
 1007 FORMAT(35H THE INITIAL STATE OF THE SYSTEM IS,/)
      PRINT 1008,(XIN(I), I=1,N)
 1008 FORMAT(6(F12.5))
      PRINT 1009
 1009 FORMAT(/,31H THE GUESSED NOMINAL CONTROL IS,/,4X,4HTIME,10X,17HCON
     1TROL VARIABLES,/)
      NSTP=NSTEP+1
      IP=NPRINT
      DO 20 I=1,NSTP
      IF (IP-NPRINT) 20,10,10
   10 T=T0+(I-1)*H
      IP=0
      PRINT 1010,T,(UNOM(J,I), J=1,NU)
 1010 FORMAT(F10.5,2(5X,F12.5))
   20 IP=IP+1
C
C     CONTANTS USED IN EXTRAPOLATION
C     AND FOR INTEGRATING
C     DX/DT=F(X,U,T) TO GET XNOM
C
```

```
      A(1)=H/2.
      A(2)=H/2.
      A(3)=H
      A(4)=H
      A(5)=H/2.
      BP(1)=2.
      BP(2)=1.
      BP(3)=1.
      BP(4)=0.
      BM(1)=0.
      BM(2)=1.
      BM(3)=1.
      BM(4)=2.
C
C     GO TO 280 AND COMPUTE XNOM
C     FROM  INTEGRATION OF DX/DT=F(X,U,T),
C     THE GRADIENT GRADJ2 AND THE
C     LOSS FUNCTION J.
C
      IND=1
      GO TO 280
C
C     PRINT XNOM,GRADJ2 AND J.PUT
C     GRADJ2 IN GRADJ1.COMPUTE
C     RIKT=-GRADJ1
C
   25 PRINT 1011
 1011 FORMAT(/,5X,6H TIME=,10X,6H XNOM=,26X,8H GRADJ2=,/)
      IP=NPRINT
      DO 40 I=1,NSTP
      IF (IP-NPRINT) 40,30,30
   30 T=T0+(I-1)*H
      IP=0
      PRINT 1012,T,(XNOM(J,I), J=1,N),(GRADJ2(K,I), K=1,NU)
 1012 FORMAT(F10.5,6(2X,F12.5),8(2X,F12.5))
   40 IP=IP+1
      PRINT 1013,SJ
 1013 FORMAT(/,5H J=,F12.5,/)
      DO 50 I=1,NSTP
      DO 50 J=1,NU
      GRADJ1(J,I)=GRADJ2(J,I)
   50 RIKT(J,I)=-GRADJ2(J,I)
C
C     COMPUTE SPROD1=<GRADJ1!GRADJ1>
C     PRINT SPROD1.
C
      DO 60 I=1,NSTP
      Q(I)=0.
      DO 60 K=1,NU
   60 Q(I)=Q(I)+GRADJ1(K,I)*GRADJ1(K,I)
      CALL QSF(H,Q,SINT,NSTP)
      SPROD1=SINT(NSTP)
      IF (SPROD1) 65,65,67
   65 PRINT 1040
 1040 FORMAT(24H SPROD1=0.END OF PROGRAM)
      CALL EXIT
   67 PRINT 1014,SPROD1
 1014 FORMAT(8H SPROD1=,F12.5)
C
C     LCOUNT=KCOUNT=NFE=0
C     LCOUNT IS COUNTER FOR LOOP
C     WHERE INTERPOLATION FAILS.
C     KCOUNT IS COUNTER FOR
C     MINIMIZATION LOOP AND NFE IS
```

```
C       COUNTER FOR HOW MANY TIMES
C       J IS COMPUTED IN THAT LOOP.
C
        KCOUNT=0
        LCOUNT=0
        NFE=0
C
C       START OF MINIMIZATION LOOP
C
   70   JB=SJ
C
C       COMPUTE MINIMIZATIONGRADIENT
C       GB=GRADJ2TRIKT
C
        DO 80 I=1,NSTP
        Q(I)=0.
        DO 80 K=1,NU
   80   Q(I)=Q(I)+GRADJ2(K,I)*RIKT(K,I)
        CALL QSF(H,Q,SINT,NSTP)
        GB=SINT(NSTP)
C
C       IF GB>0 COMPUTE RIKT=-RIKT
C       AND GB=-GB
C
        IF (GB) 110,110,90
   90   DO 100 I=1,NSTP
        DO 100 J=1,NU
  100   RIKT(J,I)=-RIKT(J,I)
        GB=-GB
        PRINT 1015
 1015   FORMAT(/,11H RIKT=-RIKT,/)
C
C       MCOUNT=0. MCOUNT IS
C       COUNTER FOR EXTRAPOLATION LOOP
C       PRINT GB.
C
  110   MCOUNT=0
        ALFA=STEP
        PRINT 1058,GB
 1058   FORMAT(4H GB=,F12.5,/)
        PRINT 1055
 1055   FORMAT(20H START-EXTRAPOLATION,/)
C
C       START EXTRAPOLATION.COMPUTE UNOM.GO TO 280
C       AND COMPUTE XNOM,GRADJ2 AND J.
C
  120   JA=JB
        GA=GB
        MCOUNT=MCOUNT+1
        PRINT 1065,ALFA
 1065   FORMAT(6H ALFA=,F12.5,/)
        DO 130 I=1,NSTP
        DO 130 J=1,NU
  130   UNOM(J,I)=UNOM(J,I)+ALFA*RIKT(J,I)
        IND=2
        GO TO 280
  140   NFE=NFE+1
C
C       PRINT XNOM AND UNOM
C
C
C       COMPUTE MINIMIZATION GRADIENT
C       GB.PRINT GB.
C
```

```
      JB=SJ
      DO 150 I=1,NSTP
      Q(I)=0.
      DO 150 K=1,NU
  150 Q(I)=Q(I)+GRADJ2(K,I)*RIKT(K,I)
      CALL QSF(H,Q,SINT,NSTP)
      GB=SINT(NSTP)
      PRINT 1049,GB
 1049 FORMAT(4H GB=,F12.5)
C
C     PRINT J.
C
      PRINT  1016,SJ
 1016 FORMAT(3H J=,F12.5,/)
C
C     INTERPOLATION OR EXTRAPOLATION?
C
      IF ((GB.GE.0).OR.(JB.GE.JA)) GO TO 160
C
C     EXTRAPOLATION.
C     PRINT ALFA.
C
  155 ALFA=4.*ALFA
      STEP=4.*STEP
      PRINT 1061,ALFA
 1061 FORMAT(6H ALFA=,F12.5,/)
      PRINT 1056
 1056 FORMAT(14H EXTRAPOLATION,/)
C
C     IF MCOUNT IS LESS THAN 20 EXTRAPOLATE
C     AGAIN ELSE STOP THE PROGRAM.
C
      IF (MCOUNT.LT.20) GO TO 120
      PRINT 1017
 1017 FORMAT(38H EXTRAPOLATION 20 TIMES.END OF PROGRAM)
      CALL EXIT
  160 CONTINUE
C
C     START INTERPOLATION.PRINT MCOUNT.
C
      PRINT 1018,MCOUNT
 1018 FORMAT(8H MCOUNT=,I2,/)
      PRINT 1057
 1057 FORMAT(14H INTERPOLATION,/)
C
C     NCOUNT=0.NCOUNT IS COUNTER FOR INTERPOLATION LOOP.
C
      NCOUNT=0
C
C     CUBIC INTERPOLATION LOOP.
C     PRINT SLAM.
C
  170 NCOUNT=NCOUNT+1
C     COMPUTE NEW UNOM.
      Z=3.*(JA-JB)/ALFA+GA+GB
      S=Z**2-GA*GB
      IF(S.LT.0.) S=0.
      W=SQRT(S)
      SLAM=ALFA*(GB+W-Z)/(GB-GA+2.*W)
      PRINT 1050,SLAM
 1050 FORMAT(6H SLAM=,F12.5)
      DO 180 I=1,NSTP
      DO 180 J=1,NU
  180 UNOM(J,I)=UNOM(J,I)-SLAM*RIKT(J,I)
```

```
C
C     GO TO 280 AND COMPUTE XNOM,GRADJ2 ANDJ.PRINT J.
C
      IND=3
      GO TO 280
  190 NFE=NFE+1
      PRINT 1019,SJ
 1019 FORMAT(3H J=,F12.5,/)
C
C     CHECK IF THE J-VALUE OF THE INTERPOLATION POINT
C     IS GREATER THAN THOSE OF THE ENDPOINTS.
C
      IF ((SJ.GT.JA.OR.SJ.GT.JB)) GO TO 210
  200 PRINT 1020
 1020 FORMAT(70H THE MINIMIZING POINT ALONG THE SEARCHDIRECTION=THE INTE
     1RPOLATED POINT,/)
      GO TO 550
  210 CONTINUE
      STEP=STEP/4.
C
C     CHECK IF THE FUNCTIONVALUE
C     OF THE RIGHT ENDPOINT IS LESS
C     THAN THAT OF THE LEFT ENDPOINT.
C
      IF(JB.LE.JA) GO TO 260
C
C     THE RIGHT ENDPOINT=THE INTERPOLATED
C     POINT.COMPUTE GB.
C     PRINT GB.
C
  220 JB=SJ
      ALFA=ALFA-SLAM
      DO 230 I=1,NSTP
      Q(I)=0.
      DO 230 J=1,NU
  230 Q(I)=Q(I)+GRADJ2(J,I)*RIKT(J,I)
      CALL QSF(H,Q,SINT,NSTP)
      GB=SINT(NSTP)
      PRINT 1060,GB
 1060 FORMAT(4H GB=,F12.5,/)
C
C     CHECK IF NCOUNT IS LESS THAN 20
C
      IF (NCOUNT.LT.20) GO TO 170
  240 CONTINUE
C
C     PRINT NCOUNT
C
      PRINT 1021,NCOUNT
 1021 FORMAT(29H INTERPOLATION FAILED NCOUNT=,I2,/)
C
C     THE MINIMIZING POINT= THE LAST
C     UNOMVALUE.SEARCHDIRECTION=-GRADJ2.
C     START TO MINIMIZE AGAIN.
C
      LCOUNT=LCOUNT+1
      DO 250 I=1,NSTP
      DO 250 J=1,NU
  250 RIKT(J,I)=-GRADJ2(J,I)
      IF (LCOUNT.LT.5) GO TO 255
      PRINT 1090
 1090 FORMAT(9H LCOUNT=5)
      CALL EXIT
  255 GO TO 70
```

```
  260 CONTINUE
C
C     THE MINIMIZING POINT=THE RIGHT
C     ENDPOINT.COMPUTE UNOM.
C
      PRINT 1095
 1095 FORMAT(40H THE MINIMIZING POINT=THE RIGHT ENDPOINT)
      DO  270 I=1,NSTP
      DO 270 J=1,NU
  270 UNOM(J,I)=UNOM(J,I)+SLAM*RIKT(J,I)
  280 CONTINUE
C
C     INTEGRATE DX/DT=F(X,U,T) TO
C     GET XNOM.RUNGE-KUTTA METHOD.
C
      DO 290 I=1,N
  290 XNOM(I,1)=XIN(I)
      T=T0
      DO 330 I=1,NSTEP
      TE=T
      DO 300 J=1,N
      XVEC(J)=XNOM(J,I)
  300 XNOM(J,I+1)=XVEC(J)
      DO 320 J=1,4
      DO 310 K=1,NU
  310 UVEC(K)=(BP(J)*UNOM(K,I)+BM(J)*UNOM(K,I+1))/2.
      CALL SYSTEM
      TE=T+A(J)
      DO 320 K=1,N
      XVEC(K)=XNOM(K,I)+A(J)*FVEC(K)
  320 XNOM(K,I+1)=XNOM(K,I+1)+A(J+1)*FVEC(K)/3.
      T=T+H
  330 CONTINUE
C
C     COMPUTE THE GRADIENT AND PUT IT IN GRADJ2.
C
C
C     PUT LAMBDA(T1) IN SLIN.
C
      TE=T1
      DO 335 J=1,N
  335 XVEC(J)=XNOM(J,NSTP)
      CALL BBOUND
      DO 340 J=1,N
  340 SLIN(J)=SFX(J)
      T=T1
C
C     LOOP WHERE THE GRADIENT=LUT+
C     FUTLAMBDA IS COMPUTED AND
C     LAMBDA BACKWARDS INTEGRATED.
C     XNOM AND UNOM ARE INTERPOLATED.GCOUNT IS COUNTER IN THAT LOOP.
C
C
C     COMPUTE THE GRADIENT AND
C     PUT IT IN GRADJ2.
C
      GCOUNT=0
  350 DO 360 J=1,N
      SLVEC(J)=SLIN(J)
  360 XVEC(J)=XNOM(J,NSTP-GCOUNT)
      DO 370 K=1,NU
  370 UVEC(K)=UNOM(K,NSTP-GCOUNT)
      TE=T
      CALL PDERFH
```

```
 1022 FORMAT(5H NFE=,I2,3X,8H NCOUNT=,I2,3X,8H LCOUNT=,I2,/)
      PRINT 1023
 1023 FORMAT(3X,6H TIME=,10X,6H XNOM=,30X,6H UNOM=,/)
      IP=NPRINT
      DO 570 I=1,NSTP
      IF (IP-NPRINT) 570,560,560
  560 T=T0+(I-1)*H
      IP=0
      PRINT 1024,T,(XNOM(J,I) ,J=1,N),(UNOM(K,I), K=1,NU)
 1024 FORMAT(F10.5,6(2X,F12.5),2(2X,F12.5))
  570 IP=IP+1
      PRINT 1041,SJ
 1041 FORMAT(3H J=,F12.5)
C
C     COMPUTE AND PRINT SPROD2.
C
      DO 580 I=1,NSTP
      Q(I)=0.
      DO 580 J=1,NU
  580 Q(I)=Q(I)+GRADJ2(J,I)*GRADJ2(J,I)
      CALL QSF(H,Q,SINT,NSTP)
      SPROD2=SINT(NSTP)
      PRINT 1025,SPROD2
 1025 FORMAT(8H SPROD2=,F12.5)
      KCOUNT=KCOUNT+1
C
C     TEST IF SPROD2<EPS
C
      IF (SPROD2-EPS) 610,610,590
  590 CONTINUE
C
C     COMPUTE BETA=SPROD2/SPROD1.PRINT BETA.
C     PUT SPROD2 IN SPROD1. COMPUTE NEW SEARCH DIRECTION.
C     PRINT RIKT.
C     PUT GRADJ2 IN GRADJ1.
C
      BETA=SPROD2/SPROD1
      PRINT 1026,BETA
 1026 FORMAT(/,6H BETA=,F12.5,/)
      SPROD1=SPROD2
      DO 600 I=1,NSTP
      DO 600 J=1,NU
      RIKT(J,I)=-GRADJ2(J,I)+BETA*RIKT(J,I)
  600 GRADJ1(J,I)=GRADJ2(J,I)
      PRINT 1052
 1052 FORMAT(6H RIKT=,/)
      DO 605 I=1,NSTP,NPRINT
  605 PRINT 1045,(RIKT(J,I) ,J=1,NU)
 1045 FORMAT(8(5X,F12.5),/)
      STEP=STEG
      GO TO 70
  610 CONTINUE
C
C     OPTIMAL SOLUTION FOUND.
C     PRINT KCOUNT,XNOM,UNOM AND J.
C
      PRINT 1027,KCOUNT
 1027 FORMAT(8H KCOUNT=,I2,/)
      PRINT 1028
 1028 FORMAT(17H OPTIMAL SOLUTION)
      PRINT 1029
 1029 FORMAT(/,3X,6H TIME=,14X,6H XNOM=,14X,6H UNOM=,/)
      IP=NPRINT
      DO 630 I=1,NSTP
```

```
      IF (IP-NPRINT) 630,620,620
  620 T=T0+(I-1)*H
      IP=0
      PRINT 1030,T,(XNOM(J,I), J=1,N),(UNOM(K,I), K=1,NJ)
 1030 FORMAT(F10.5,6(2X,F12.5),2(2X,F12.5))
  630 IP=IP+1
      PRINT 1031,SJ
 1031 FORMAT(3H J=,F12.5)
      CALL EXIT
      END
```

COMPILATION: NO DIAGNOSTICS.