STRAFF-FUNKTIONSMETODER FÖR NUMERISK LÖSNING AV OPTIMALA STYRPROBLEM

ROLF PERSSON

RE-144 Juli 1974
Inst. för Reglerteknik
Lunds Tekniska Högskola

STRAFF-FUNKTIONSMETODER FÖR NUMERISK LÖSNING AV OPTIMALA STYRPROBLEM.

ROLF PERSSON

RE-144 juli 1974

Inst. för Reglerteknik

Lunds Tekniska Högskola

STRAFF-FUNKTIONSMETODER FÖR NUMERISK LÖSNING AV OPTIMALA STYRPROBLEM.

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

1. ABSTRACT.

In this thesis a computer program(IPF1) for numerical solution of optimal control problems is presented.
The program is based on a method that uses a " Penalty-function",which is to be minimized.
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 examensarbete presenteras ett datorprogram(IPF1) för numerisk lösning av optimala styrproblem.
programmet är baserat på en metod som utnyttjar en "Penalty-function" (straff-funktion),som skall minimeras.
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 m.

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

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

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

Metod: Bivillkoret $\dot{x}(t)=f(x(t),u(t))$ tages om hand genom att minimera en "Penalty-function"

$$P(\dot{x}(t);u(t))=F(x(t_1)+\int_{t_0}^{t_1} L(x(t);u(t))dt+C\cdot\langle f(x(t),u(t))-\dot{x}(t);f(x(t),u(t))-\dot{x}(t)\rangle$$

m.a.p. $\dot{x}(t)$ och u(t). Observera att x(t) då ges av $x(t)=x_0+\int_{t_0}^{t_1}\dot{x}(s)ds$. Genom att göra talet C stort kommer vid minimum för P bivillkoret $\dot{x}(t)=f(x(t),u(t))$ att ungefär vara uppfyllt och om den optimala lösningen är x^* och u^* kommer $P(x^*,u^*)\approx J(\hat{x},\hat{u})$ där $\hat{x}$ och $\hat{u}$ är de värden som minimerar J under bivillkoret $\dot{x}(t)=f(x(t),u(t))$. För de matematiska bevisen och för grundligare studier hänvisas till [3]. Man kan visa att i vissa fall så gäller att $\lim_{C\to\infty}(x^*,u^*)=(\hat{x},\hat{u})$ och $\lim_{C\to\infty}P(x^*,u^*)=J(\hat{x},\hat{u})$.

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. $\langle f(x,u,t)-\dot{x}(t),f(x,u,t)-\dot{x}(t)\rangle$

är en skalärprodukt och kan skrivas

$$\int_{t_o}^{t_1} (f(x,u,t)-\dot{x}(t))^T (f(x,u,t)-\dot{x}(t))dt$$

4. Teori.

Jag använder mig av Fletcher-Reeves metod, vilken är en konjugerad gradientmetod, för att numeriskt minimera P. För att kunna använda denna metod behöver man kunna beräkna gradienten av P m.a.p. u och $\dot{x}$.

Beräkning av gradienten m.a.p. u.

Vi definierar differentialen av P m.a.p. u med inkrementet h som:

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

om δP_u existerar och är linjär och kontinuerlig i h.

Beräkning av denna differential:

$$P(\dot{x},u+\varepsilon h)=F(x(t_1))+\int_{t_o}^{t_1} L(x,u+\varepsilon h)dt+C\cdot\langle f(x,u+\varepsilon h)-\dot{x},f(x,u+\varepsilon h)-\dot{x}\rangle$$

Taylorutveckling av L och f kring u ger:

$$P(\dot{x},u+\varepsilon h)=F(x(t_1))+\int_{t_o}^{t_1}\{L+L_u\cdot\varepsilon\cdot h+\mathcal{O}(\varepsilon)\}dt+C\cdot\langle f+f_u\cdot\varepsilon\cdot h+\mathcal{O}(\varepsilon)-\dot{x},$$

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

Vi får då

$$\frac{P(\dot{x},u+\varepsilon h)-P(\dot{x},u)}{\varepsilon}=\int_{t_o}^{t_1}\left\{L_u\cdot h+\frac{\mathcal{O}(\varepsilon)}{\varepsilon}\right\}dt+C\cdot 2\langle f,f_u\cdot h\rangle-C\cdot 2\langle\dot{x},f_u\cdot h\rangle+\frac{\mathcal{O}(\varepsilon)}{\varepsilon}$$

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

$$\delta P_u(\dot{x},u,h)=\int_{t_o}^{t_1}L_u h\,dt+2\cdot C\cdot\langle f-\dot{x},f_u\cdot h\rangle \qquad (1)$$

Enligt ovan har vi definierat skalärprodukten som

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

Vi kan då skriva (1) som

$$\delta P_u(\dot{x},u,h)=\langle\{L_u^T+2\cdot C\cdot f_u^T(f-\dot{x})\},h\rangle \qquad (2)$$

Om $\delta P_u(\dot{x},u,h)$ kan skrivas $\delta P_u=\langle\nabla_u P,h\rangle$ så definierar vi gradienten av P m.a.p. u som $\nabla_u P$. Genom att jämföra med (2)

fås

$$\nabla_u P = L_u^T + 2 \cdot C \cdot f_u^T (f - \dot{x})$$

Beräkning av P m.a.p. $\dot{x}$.

För att beräkna gradienten av P m.a.p. $\dot{x}$ måste vi taga hänsyn till att x bestäms ur sambandet

$$x(t) = x_o + \int_{t_o}^{t_1} \dot{x}(s)\,ds \ .$$

Ger vi alltså $\dot{x}(t)$ ett tillskott $\dot{h}(t)$ kommer x(t) att få tillskottet $\int_{t_o}^{t_1} \dot{h}(s)\,ds = h(t) - h(t_o) = h(t)$ eftersom $h(t_o) = 0$, dvs.

x_o är fixt.

Vi kan då beräkna differentialen av P m.a.p. $\dot{x}$ med inkrementet $\dot{h}$ som

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

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

Vi har (kom ihåg att x övergår i $x + \varepsilon h$ med $h(t_o) = 0$)

$$P(\dot{x} + \varepsilon \dot{h}, u) = F(x(t_1) + \varepsilon h(t_1)) + \int_{t_o}^{t_1} L(x + \varepsilon h, u)\,dt + C \cdot \langle f(x + \varepsilon h, u) - \dot{x}$$

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

Taylorutveckla F, L och f kring x och skriv skalärprodukten $\langle \xi, \eta \rangle$ som $\int_{t_o}^{t_1} \xi^T \eta \, dt$.

$$P(\dot{x} + \varepsilon \dot{h}, u) = F(x(t_1)) + F_x \cdot \varepsilon \cdot h(t_1) + \mathcal{O}(\varepsilon) + \int_{t_o}^{t_1} \{ L + L_x \cdot \varepsilon \cdot h + \mathcal{O}(\varepsilon) +$$

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

$$= F(x(t_1)) + F_x \cdot \varepsilon \cdot h(t_1) + \mathcal{O}(\varepsilon) + \int_{t_o}^{t_1} \{ L + L_x \cdot \varepsilon \cdot h + C \cdot (f - \dot{x})^T (f - \dot{x}) +$$

$$2 \cdot C \cdot (f^T f_x \cdot \varepsilon \cdot h - \dot{x}^T f_x \cdot \varepsilon \cdot h + f^T \cdot \varepsilon \cdot \dot{h} + \dot{x}^T \cdot \varepsilon \cdot \dot{h}) + \mathcal{O}(\varepsilon) \} dt$$

Efter lite räkningar fås

$$\delta P_{\dot{x}}(\dot{x},u,\dot{h}) = F_x h(t_1) + \int_{t_o}^{t_1} \{L_x + 2 \cdot C \cdot (f-\dot{x})^T f_x\} h dt - \int_{t_o}^{t_1} 2 \cdot C \cdot (f-\dot{x})^T \dot{h} dt \qquad (3)$$

Den första termen kan skrivas om

$$\int_{t_o}^{t_1} F_x(x(t_1))\dot{h}(t_1)dt$$

ty $h(t_o)=0$

Den andra termen skrivs om med hjälp av följande lemma.

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

$$\int_{t_o}^{t_1} \xi^T(t)h(t)dt = \int_{t_o}^{t_1} \left(\int_{t_o}^{t_1} \xi^T(s)ds \right) \dot{h}(t)dt$$

för alla $\xi(t)$.

Den andra termen i (3) blir då

$$\int_{t_o}^{t_1} \left\{ \int_{t}^{t_1} \left[L_x + 2 \cdot C \cdot (f-\dot{x})^T f_x \right] ds \right\} \dot{h}(t)dt$$

och vi får

$$\delta P_{\dot{x}}(\dot{x},u,\dot{h}) = \int_{t_o}^{t_1} F_x(x(t_1))\dot{h}(t)dt + \int_{t_o}^{t_1} \{-2 \cdot C \cdot (f-\dot{x})^T +$$

$$\int_{t}^{t_1} \left[L_x + 2 \cdot C \cdot (f-\dot{x})^T f_x \right] ds \} \dot{h}(t)dt$$

Alltså kan $\delta P_{\dot{x}}$ skrivas som $\delta P_{\dot{x}} = \langle \nabla_{\dot{x}} P, \dot{h} \rangle$ och sålunda fås

$$\nabla_{\dot{x}} P = F_x^T(x(t_1)) - 2 \cdot C \cdot (f-\dot{x}) + \int_{t}^{t_1} \left[L_x^T + 2 \cdot C \cdot f_x^T (f-\dot{x}) \right] ds$$

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 algorit-

men med de modifieringar som gjorts.

Fletcher-Reeves metod för ändligtdimensionella problem:

1. Beräkna gradienten $g_o=\nabla f(x_o)^T$ för givet x_o och sätt begynnelseriktning $d_o=-g_o$.

2. För $k=0,1,\ldots,n-1$

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 $d_{k+1}=-g_{k+1}+\beta_k d_k$

där $\beta_k = \dfrac{g_{k+1}^T \cdot g_{k+1}}{g_k^T \cdot g_k}$

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

Modifierad Fletcher-Reeves för styrproblemet.

1. Beräkna gradienten $g_o=\nabla P(\xi_o)^T$ för givet ξ_o och sätt begynnelseriktning $d_o=-g_o$.

2.a) Sätt $\xi_{k+1}=\xi_k+\alpha_k d_k$ där α_k minimerar $P(\xi_k+\alpha d_k)$.

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

c) Om $\langle g_{k+1}, g_{k+1}\rangle \equiv \int_{t_o}^{t_1} g_{k+1}^T g_{k+1}\,dt > \varepsilon$

sätt ny sökriktning $d_{k+1} = -g_{k+1} + \beta_k d_k$ där $\beta_k = \dfrac{g_{k+1}^T \cdot g_{k+1}}{g_k^T \cdot g_k}$

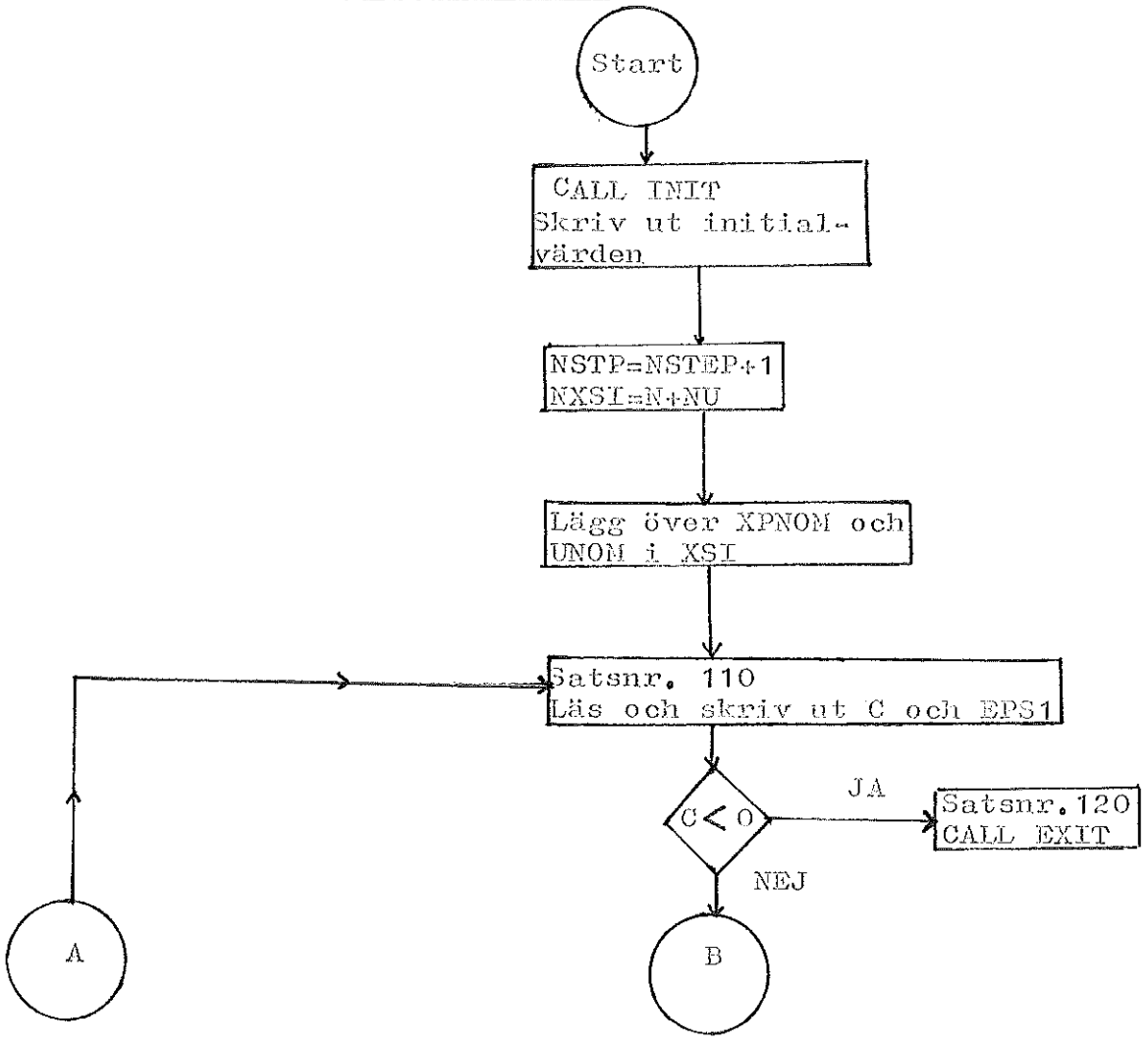
gå till 2a). Om $\langle g_{k+1}, g_{k+1}\rangle < \varepsilon_1$ är minimeringen för det aktuella värdet på C färdig.

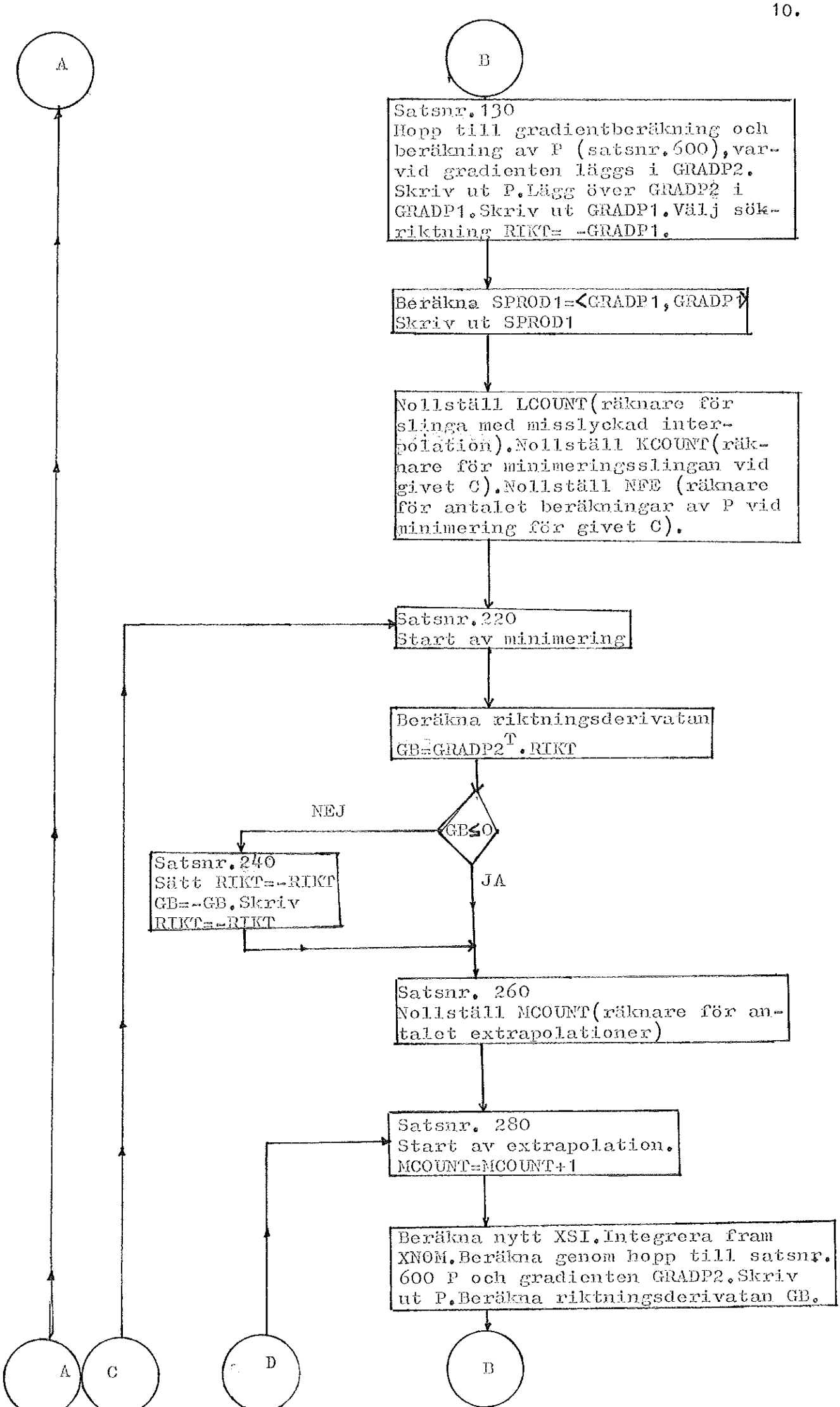
Flödesschema för den modifierade varianten av Fletcher-Reeves finns i appendix A.

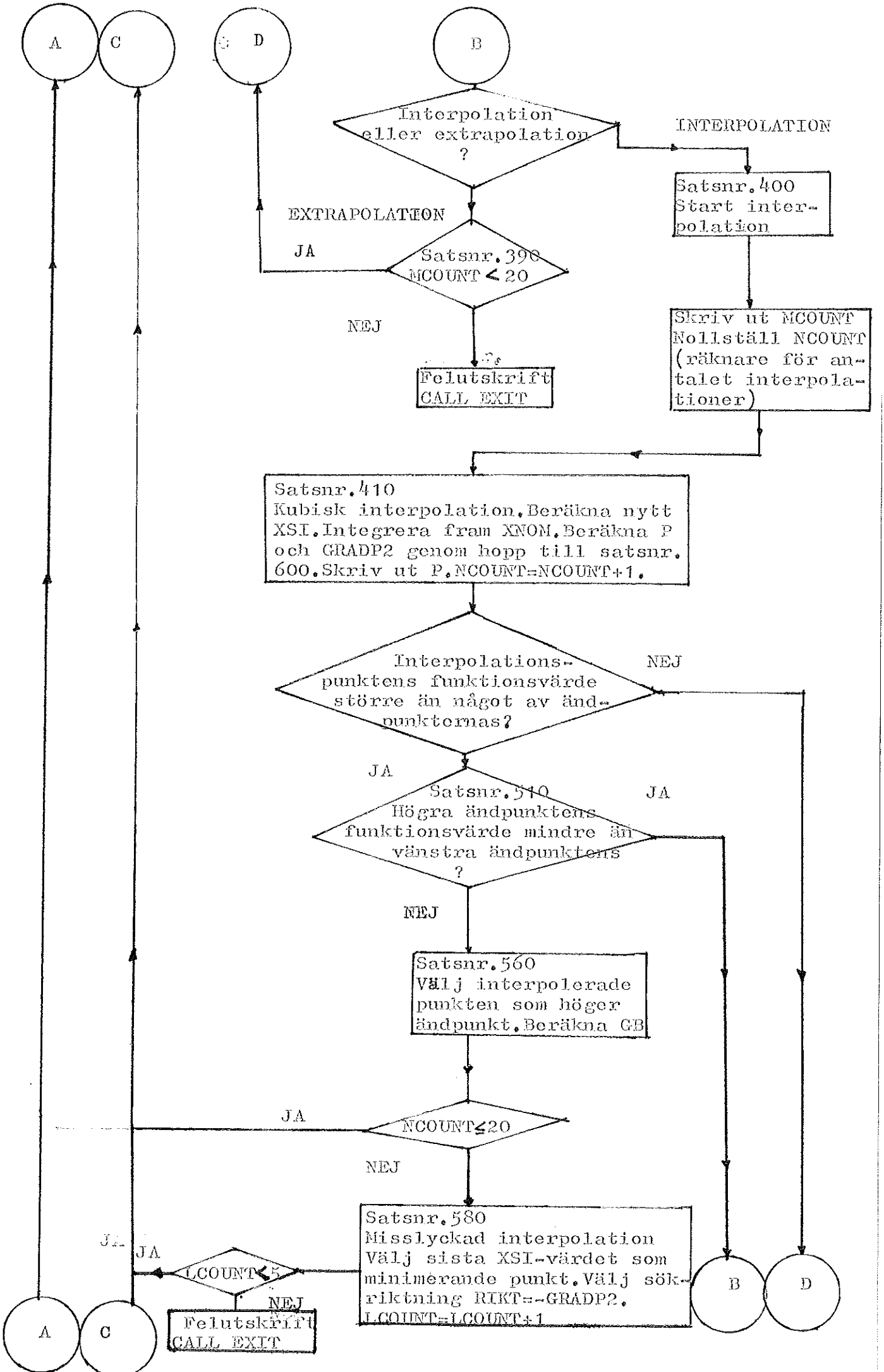
5. Flödesschema för program IPF1.

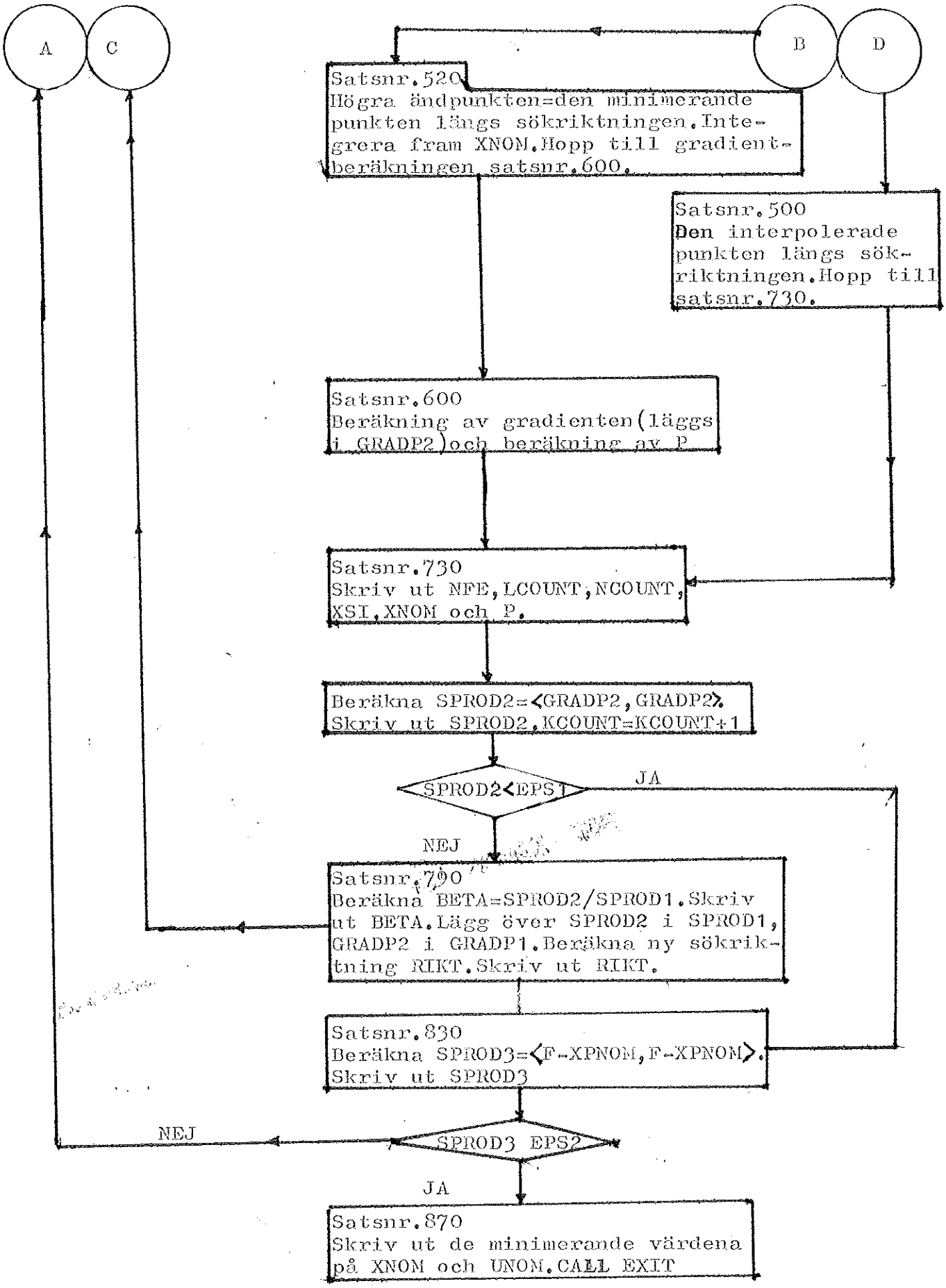
Nedan anges flödesschemat för program IPF1. Det baserar sig på den modifierade varianten av Fletcher-Reeves, som beskrivs i kap. 4 och vars flödesschema finns i appendix A.
Flödesschemat visar i stort hur programmet fungerar utan att gå in på själva beräkningarna, vilka istället beskrivs i kap. 7, 8, 9, 10. Däremot betonas mer utskrifter och hopp, varför satsnummer från programmet har satts ut i flödesschemat vid hopp.
För förklarande av symboler m.m. och förtydligande av programmet se kap. 6.

Program IPF1.




A
B
Satsnr.130
Hopp till gradientberäkning och beräkning av P (satsnr.600),varvid gradienten läggs i GRADP2. Skriv ut P.Lägg över GRADP2 i GRADP1.Skriv ut GRADP1.Välj sökriktning RIKT= -GRADP1.
Beräkna SPROD1=<GRADP1,GRADP1>
Skriv ut SPROD1
Nollställ LCOUNT(räknare för slinga med misslyckad interpolation).Nollställ KCOUNT(räknare för minimeringsslingan vid givet C).Nollställ NFE (räknare för antalet beräkningar av P vid minimering för givet C).
Satsnr.220
Start av minimering
Beräkna riktningsderivatan GB=GRADP2^T.RIKT
GB≤0
NEJ
JA
Satsnr.240
Sätt RIKT=-RIKT
GB=-GB.Skriv RIKT=-RIKT
Satsnr. 260
Nollställ MCOUNT(räknare för antalet extrapolationer)
Satsnr. 280
Start av extrapolation.
MCOUNT=MCOUNT+1
Beräkna nytt XSI.Integrera fram XNOM.Beräkna genom hopp till satsnr. 600 P och gradienten GRADP2.Skriv ut P.Beräkna riktningsderivatan GB.
A
C
D
B


A
C
D
B
Interpolation eller extrapolation ?
INTERPOLATION
EXTRAPOLATION
Satsnr.400 Start interpolation
JA
Satsnr.390 MCOUNT < 20
NEJ
Felutskrift CALL EXIT
Skriv ut MCOUNT Nollställ NCOUNT (räknare för antalet interpolationer)
Satsnr.410 Kubisk interpolation.Beräkna nytt XSI.Integrera fram XNOM.Beräkna P och GRADP2 genom hopp till satsnr. 600.Skriv ut P.NCOUNT=NCOUNT+1.
Interpolationspunktens funktionsvärde större än något av ändpunktemas?
NEJ
JA
Satsnr.510 Högra ändpunktens funktionsvärde mindre än vänstra ändpunktens ?
JA
NEJ
Satsnr.560 Välj interpolerade punkten som höger ändpunkt.Beräkna GB
JA
NCOUNT≤20
NEJ
Satsnr.580 Misslyckad interpolation Välj sista XSI-värdet som minimerande punkt.Välj sökriktning RIKT=-GRADP2. LCOUNT=LCOUNT+1
JA
LCOUNT<5
NEJ
Felutskrift CALL EXIT
A
C
B
D


A
C
B
D
Satsnr.520
Högra ändpunkten=den minimerande punkten längs sökriktningen.Integrera fram XNOM.Hopp till gradientberäkningen satsnr.600.
Satsnr.500
Den interpolerade punkten längs sökriktningen.Hopp till satsnr.730.
Satsnr.600
Beräkning av gradienten(läggs i GRADP2)och beräkning av P
Satsnr.730
Skriv ut NFE,LCOUNT,NCOUNT, XSI,XNOM och P.
Beräkna SPROD2=<GRADP2,GRADP2>
Skriv ut SPROD2,KCOUNT=KCOUNT+1
SPROD2<EPS1
JA
NEJ
Satsnr.790
Beräkna BETA=SPROD2/SPROD1.Skriv ut BETA.Lägg över SPROD2 i SPROD1, GRADP2 i GRADP1.Beräkna ny sökriktning RIKT.Skriv ut RIKT.
Satsnr.830
Beräkna SPROD3=<F-XPNOM,F-XPNOM>.
Skriv ut SPROD3
NEJ
SPROD3 EPS2
JA
Satsnr.870
Skriv ut de minimerande värdena på XNOM och UNOM.CALL EXIT

6. Förtydligande och förklarande av programmet

I programmet approximeras tidfunktionerna (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 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. Sista index hos dessa fält anger alltid tidpunkten och betecknas med I. I sättes lika med 1 vid starttidpunkten TO (=to) och lika med NSTP = NSTEP + 1 vid sluttidpunkten T1 (=t1). I anger den aktuella tidpunkten t genom sambandet T=TO + (I-1)·H, där H är intervallängden. Nedan anges sambandet mellan olika vektorer och motsvarande fält.

$x(t) \longleftrightarrow$ XNOM (K,I) K = 1,...,N

$u(t) \longleftrightarrow$ UNOM (J,I) J = 1,...,NU

$\dot{x}(t) \longleftrightarrow$ XPNOM (K,I) K = 1,...,N

$\xi = \begin{pmatrix} \dot{x} \\ u \end{pmatrix} \longleftrightarrow$ XSI (K,I) K = 1,...,NXSI

sökriktning $d(t) \longleftrightarrow$ RIKT (K,I) K = 1,...,NXSI

gradienten av P m.a.p. $\dot{x}$ och $u = g(t) \longleftrightarrow$ GRADP1 (K,I) eller GRADP2 (K,I) K = 1,...,NXSI

Vidare användes följande beteckningar

$n \longleftrightarrow$ N

$m \longleftrightarrow$ NU

$n+m \longleftrightarrow$ NXSI

$x(t_o) \longleftrightarrow$ XIN(K) K = 1,...,N

$F(x(t_o)) \longleftrightarrow$ SF

$f^k \longleftrightarrow$ F (K) K = 1,...,N $\quad f = \begin{bmatrix} f^1 \\ f^n \end{bmatrix}$

$f^k_x \longleftrightarrow$ FX (K,L) K = 1,...,N L = 1,...,N

$\left(\frac{\partial f^k}{\partial x_L} = \text{FX (K,L)} \right)$

f_u^j ⟷ FU (K,J) K = 1,...,. N J = 1,..., NU

$\left(\frac{\partial f^k}{\partial u_j} \right.$ = FU (K,J))

L ⟷ SL

L_x ⟷ SLX(K) K = 1,..., N

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

L_U ⟷ SLU(J) J = 1,..., NU

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

t_o ⟷ TO

t_1 ⟷ T1

$\langle g_{k+1}, g_{k+1} \rangle$ ⟷ SPROD1 eller SPROD2

$\langle f_{-x}, f_{-\dot{x}} \rangle$ ⟷ SPROD3

β ⟷ BETA

ε_1 ⟷ EPS1

Varje gång XSI ändras i programmet beräknas nytt XNOM enligt

$x(t) = x_o + \int_{t_o}^{t} \dot{x}(s)ds$.

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 Yi är givna vid ekvidistanta punkter $x_i = a + (i-1).h$ beräknar QSF fältet z där $z_i = \int_{a}^{x_i} y(x)dx$, i = 1,...,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). För närmare studier av subrutinen hänvisas till [5].

Beräkning av gradienten av P m.a.p. $\dot{x}$ och u och beräkning av funktionsvärdet av P görs på ett ställe i programmet och utnyttjas genom hopp från de ställen i programmet, där dessa värden behöver uträknas. Genom väljarstyrt hopp sker återhopp till rätt ställe.

Programmets funktion i stort: Programmet för först initialvärden från subrutinen USER (se närmare kap. 8). Sedan läses värdena på C och EPS1 från datakort, varefter minimeringen för detta C-värde börjar.

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 (SPROD 2) har blivit tillräckligt liten (mindre än EPS1). Om så är fallet betyder det att man ligger på minimum varför minimeringen för detta C avbryts. I annat fall väljs en ny sökriktning, som man minimerar längs och så fortsätter programmet till SPROD 2 har blivit mindre än EPS1.

När programmet minimerat färdigt för ett C testas om ekvationen $\dot{x} = f$ är uppfylld. Detta görs genom att beräkna skalärprodukten av $f-\dot{x}$ med sig själv (SPROD 3). Om denna skalärprodukt är liten (mindre än EPS2) är ekvationen ungefär uppfylld och programmet skriver ut de minimerande x- och u-värdena. Annars läses nya C och EPS1 och minimering för det nya C-värdet börjar varvid de x- och u-värden som minimerade för förra C-värdet används som startvärden. Programmet kommer att minimera för nya C-värden tills att antingen SPROD3 är tillräckligt liten eller att ett negativt C-värde påträffas (ligger sist bland datakorten) varvid programmet avbryts.

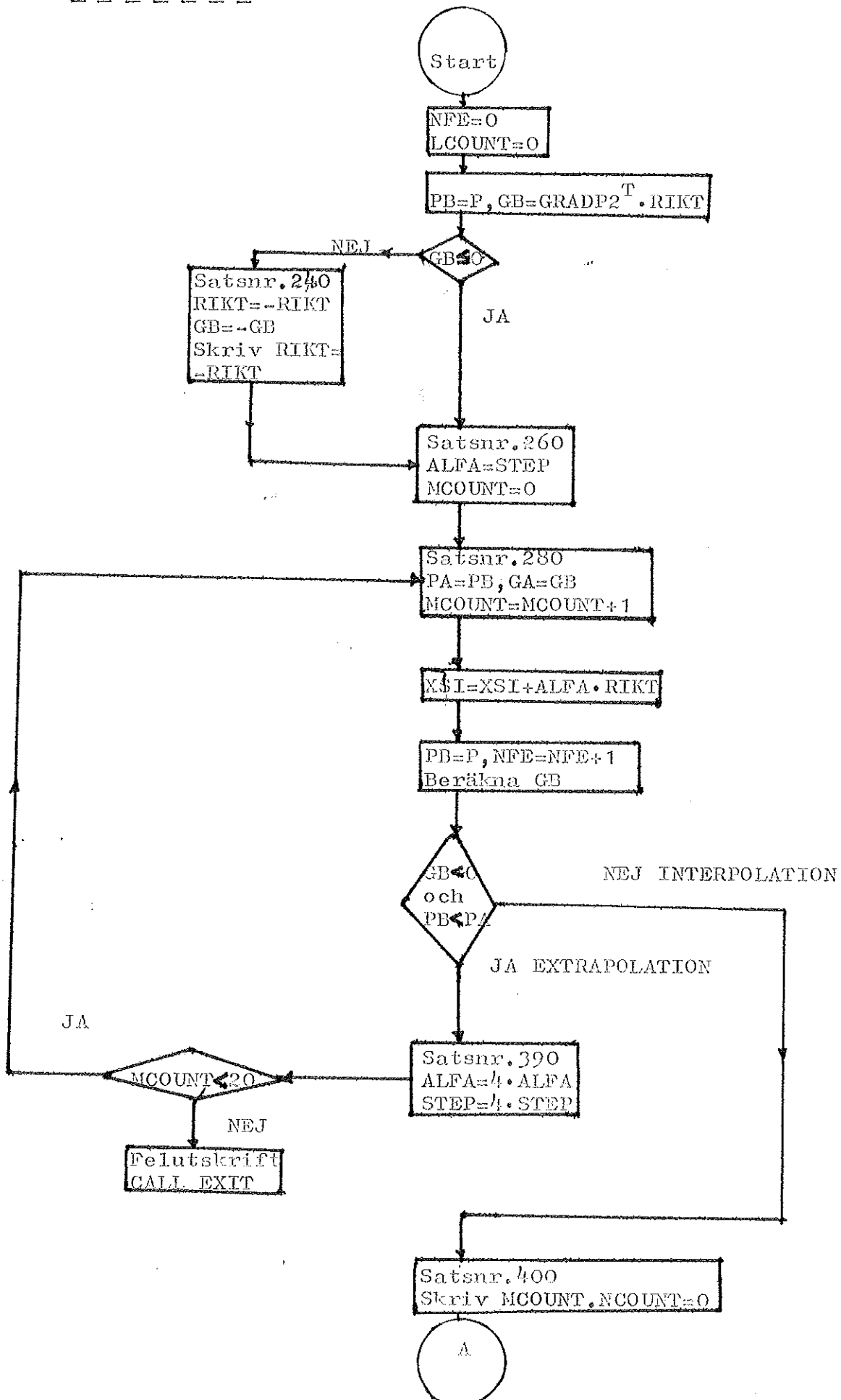
Utskrifter: Först skrivs en del initialvärden ut såsom N, NU,, TO, T1, H, NSTEP, NPRINT, (se kap. 8),. XIN, gissade startvärden UNOM, XPNOM och det därvid framintegrerade XNOM. I huvudloopen där minimering för bestämt C sker skrivs först ut de aktuella värdena på C, EPS1 och funktionsvärdet av P. Därefter skrivs gradienten GRADP1 och dess skalärprodukt med sig själv SPROD1 ut. Detta görs bara en gång i huvudloopen, nämligen vid första sökriktningen som sätts lika med negativa gradienten.

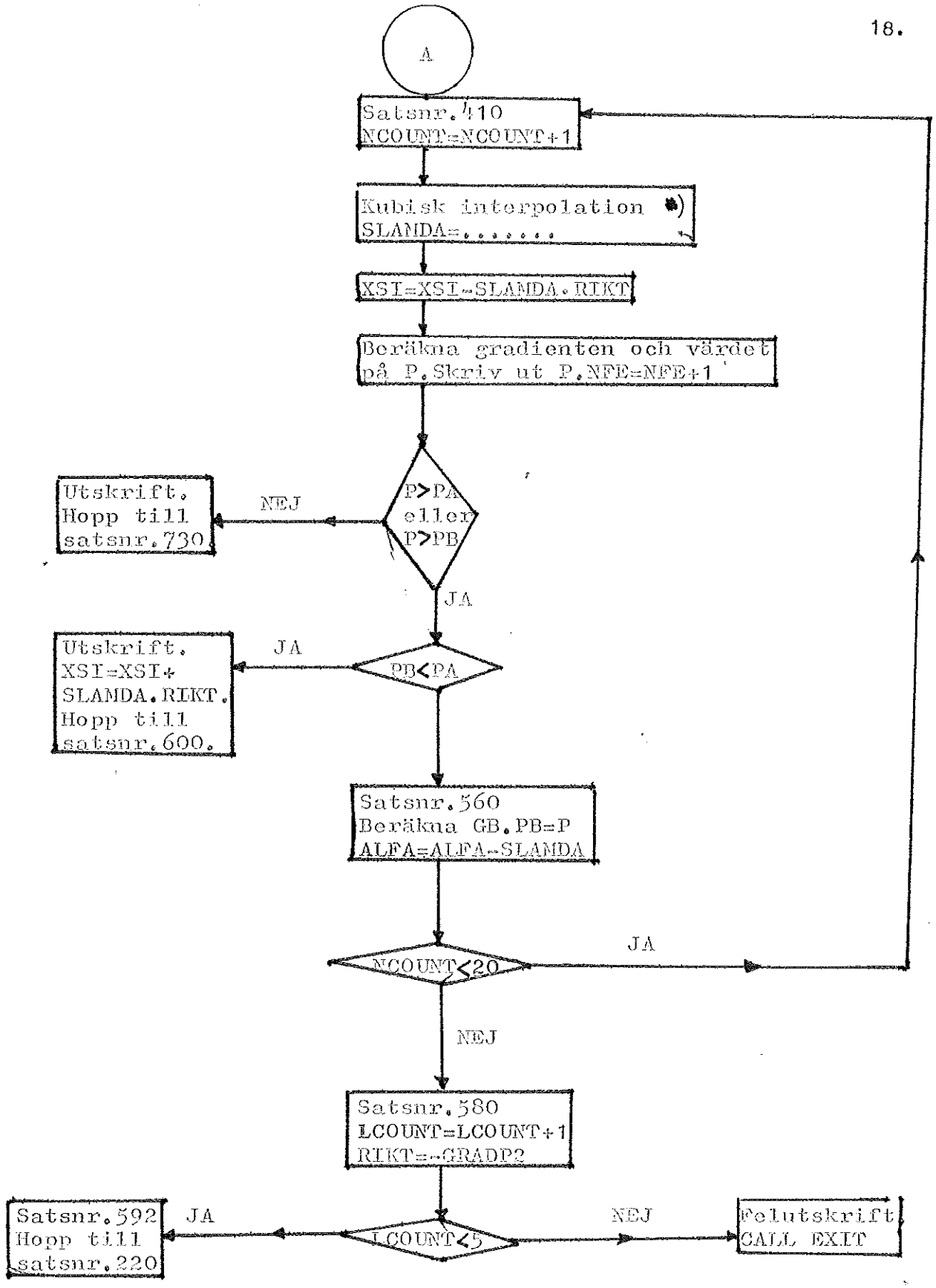
Inuti minimeringsslingan för en speciell riktning skrivs först ut vilka funktionsvärden P får vid extrapolation och antalet extrapolationer (MCOUNT). Därefter skrivs ut vad P får för värde vid interpolation, vilken punkt, som har tagits som

minimerande punkt, och antalet interpolationer (NCOUNT). Mer om utskrifter i minimeringsslingan kan läsas i kap. 7.
Efter minimeringsslingan skrivs en del räknare och de minimerande värdena på XSI och XNOM för den aktuella riktningen ut (se kap.7). Därefter skrivs SPROD2 ut och om denna inte är tillräckligt liten, skrivs BETA = (skalärprodukten av nya gradienten medsig själv (SPROD2)) (Skalärprodukten av den gamla gradienten med sig själv) och den nya riktningen RIKT ut. Om däremot SPROD2 är tillräckligt liten, skrivs antalet riktningar (KCOUNT), som programmet har minimerat längs för det aktuella C-värdet, och SPROD3 ut. Om SPROD3 inte är tillräckligt liten startar huvudloopen om igen annars skrivs de optimala värdena på XNOM och UNOM ut.

7. Beskrivning av den endimensionella minimeringen.

Flödesschema.





*) $Z=3\cdot(PA-PB)/ALFA+GA+GB$

$W=\sqrt{Z^2-GA\cdot GB}$

$SLAMDA=ALFA\cdot(GB+W-Z)/(GB-GA+2\cdot W)$

Satsnummer har hämtats från programmet.

Förklaring av symboler:

NFE= antal funktionsberäkningar.
STEP= steglängd, som används vid nästa minimering.
ALFA= steglängd vid extrapolation.
MCOUNT= antalet extrapolationer vid en minimering.
NCOUNT= antalet interpolationer vid en minimering.
P= aktuellt funktionsvärde.
PA= funktionsvärde i vänstra punkten.
PB= funktionsvärde i högra punkten.
GB= riktningsderivatan= $GRADP2^T \cdot RIKT$.
GRADP2= gradienten i aktuell punkt.
RIKT= aktuell sökriktning.
XSI= $\begin{bmatrix} \hat{x} \\ u \end{bmatrix}$

Funktion: Vi minimerar längs en riktning RIKT, genom att ändra på XSI längs denna riktning enligt XSI = XSI + ALFA · RIKT. Metoden börjar med att beräkna funktionsvärdet och minimeringsgradienten för givet XSI (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" skrivs ut.

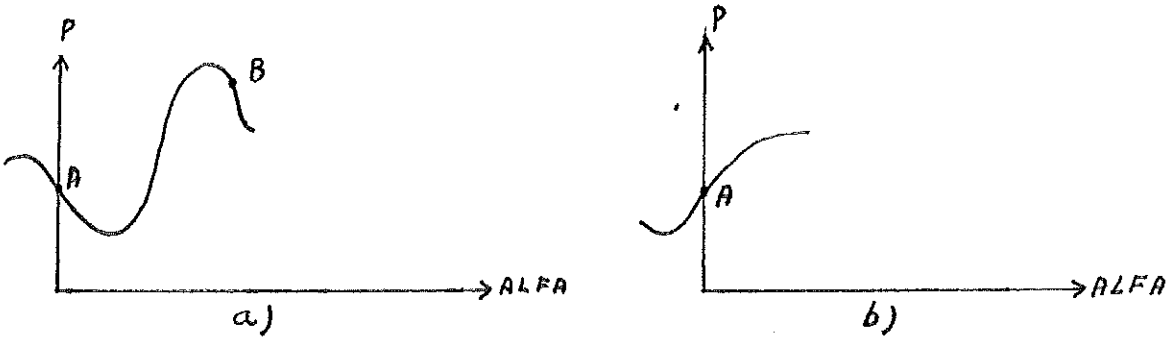
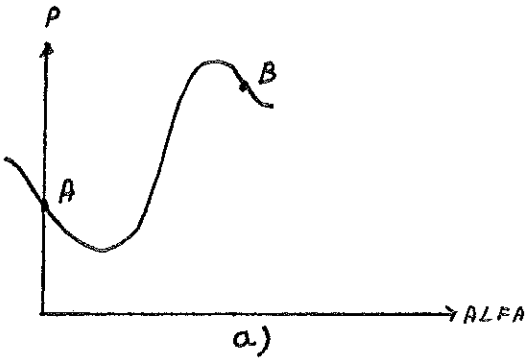


fig. 1a) Negativ riktningsderivata
fig. 1b) Positiv riktningsderivata

Därefter görs en extrapolation med steglängden ALFA och funktionsvärdet och riktningsderivata beräknas i denna punkt (punkt B). Om PA $<$ PB (fig. 2a) eller GB $>$ 0 (fig. 2b), så startas interpolationen, i annat fall multipliceras steglängden med fyra och ny extra-

polation sker.



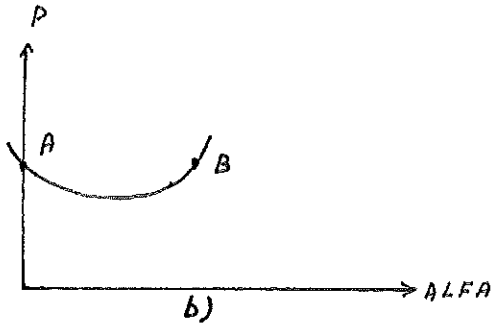


fig. 2a) PA < PB

fig. 2b) GB > 0

Om 20 extrapolationer gjorts i rad, så avbryts programmet och "MCOUNT GREATER THAN 20" skrivs ut.
Vid interpolationens början skrivs ut antalet gånger extrapolation har gjorts "MCOUNT =". Sedan görs en kubisk interpolation, varvid man anpassar ett tredjegradspolynom till kurvan. Därvid beräknas hjälpstorheterna Z, W samt SLAMDA, som anger hur långt steg tillbaka man skall ta från den högra punkten vid interpolation (se fig. 3).

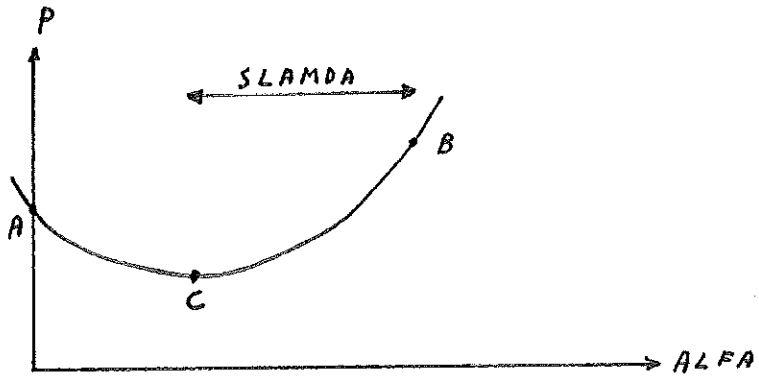


fig. 3) kubisk interpolation

I den nya punkten (punkt C) beräknas funktionsvärdet och om detta är mindre än ändpunkternas funktionsvärden PA och PB, så är minimeringen färdig (se fig. 4a) och "CHOOSE THE INTERPOLATED POINT AS THE MINIMIZING POINT ALONG THE SEARCHDIRECTION RIKT" skrivs ut. Om så inte är fallet divideras steglängden med 4, och om PB < PA väljes B som minimerande punkt och "CHOOSE THE RIGHT POINT AS THE MINIMIZING POINT ALONG THE SEARCHDIRECTION

RIKT" skrivs ut (se fig. 4b). Om PA $<$ PB och PA $<$ PC väljes den interpolerade punkten som höger ändpunkt B och ny interpolation görs (se fig. 4c).

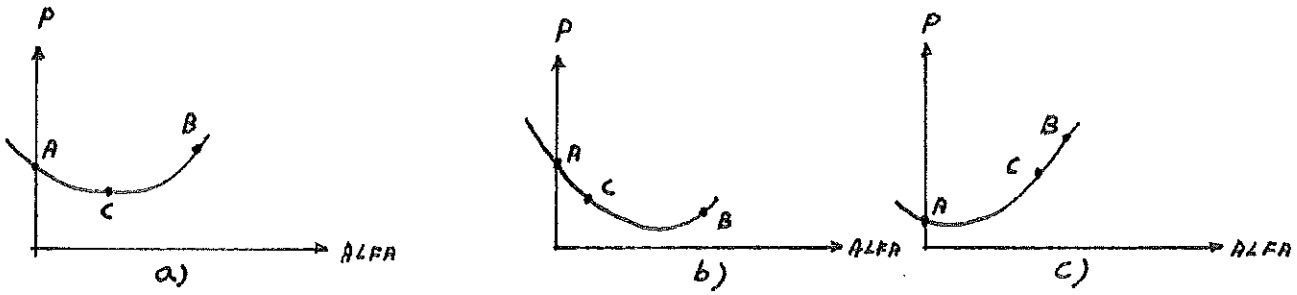


fig. 4a) PC $<$ PA och PC $<$ PB
fig. 4b) PB $<$ PA och PC $>$ PB
fig. 4c) PA $<$ PB och PA $<$ PC

Om 20 interpolationer görs i sträck avbryts minimeringen och omstart sker av FLETCHER-REEVES med sökriktning lika med negativa gradienten i den aktuella punkten. "NCOUNT GREATER THAN 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 kubiska inter polationen kan uttrycket Z^2 - GA · GB, vilket står under ett rottecken, bli negativt, varför Z^2 - GA · GB i så fall sättes lika med noll. Efter minimeringen skriver programmet ut "VALUES WHEN OUT OF THE MINIMISATION LOOP FOR THE SEARCHDIRECTION RIKT" och värdena på NFE" LCOUNT" NCOUNT" XSI och XNOM skrivs ut. LCOUNT är en räknare, som anger antalet omstarter då mer än 20 interpolationer gjorts i rad. Om mer än 5 sådana omstarter gjorts avbryts programmet och "LCOUNT GREATER THAN 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, F. När USER beräknar 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 består av ett antal underavdelningar med olika ENTRY POINTS, vilka kan anropas var för sig med en CALL-instruktion. Vi kommer nedan att 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 TO (starttidpunkt) till T1 (sluttidpunkt) har indelats i, varvid intervallängden H blir $\frac{T1-TO}{NSTEP}$. Obs. Glöm ej att ändra H 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. Begynnelsevärden på UNOM, XPNOM, dvs. initialgissningen, placeras också här. EPS2 anger hur liten skalärprodukten av (F-XPNOM) med sig själv skall vara, för att hela programmet skall vara färdigt. Skalärprodukten blir således ett mått på hur bra ekvationen $\dot{x} = f$ är uppfylld.
STEP är steglängden vid extrapolation första gången vi minimerar för givet C och vid omstart (NCOUNT > 20). Sedan modifieras STEP i minimeringen.

ENTRY CPARAM: Vid anrop av denna avdelning läses nytt

värde på C och om detta värde är negativt avbryts programmet. Vidare läses EPS1 som anger hur liten skalärprodukten av gradienten med sig själv skall vara för att minimeringen för ett givet C skall vara färdig.

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

ENTRY INTLOS: SL är den funktion i P, som finns under integralstecknet.

ENTRY B BOUND:Här finns de partiella derivatorna av SF m.a.p. X.

ENTRY PDER: 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.

Gradienten av P m.a.p. XSI består av två delar dels m.a.p. $\mathring{x}$ och dels m.a.p. u .

a) Gradienten m.a.p. u.

Teorin enl. kap.4 ger: $\nabla_u P = L_u^T + 2 \cdot C \cdot f_u^T (f - \mathring{x})$.
$\nabla_u P$ är en kolonnvektor med m (=NU) komponenter. L_u är en radvektor med m komponenter medan f_u är en n·m (N·NU) matris. f och $\mathring{x}$ är kolonnvektorer med n (N) komponenter.
Observera att varje komponent är en funktion.
För en bestämd tidpunkt blir

$$ {}_u P\,(t) = \begin{bmatrix} L_{u_1} + 2 \cdot C \cdot \sum_{i=1}^{n} \left(f^i_{u_1} \cdot (f_i - \mathring{x}_i) \right) \\ \vdots \\ L_{u_m} + 2 \cdot C \cdot \sum_{i=1}^{n} \left(f^i_{u_m} \cdot (f_i - \mathring{x}_i) \right) \end{bmatrix} $$

Med utgångspunkt från denna vektor har nedanstående programavsnitt skrivits.

```
C       START COMPUTE GRADIENT UNOM
        DO 720 I=1,NSTP
        TE=T0+(I-1)*H
        DO 690 L=1,N
  690   XVEC(L)=XNOM(L,I)
        DO 700 K=1,NU
        N1=N+K
  700   UVEC(K)=XSI(N1,I)
        CALL PDER
        CALL SYSTEM
        DO 720 J=1,NU
        SUM=0.
        DO 710 L=1,N
  710   SUM=SUM+FU(L,J)*(F(L)-XSI(L,I))
        K=J+N
  720   GRADP2(K,I)=SLU(J)+2.*C*SUM
C
C       END GRADIENT UNOM
```

b) Gradienten av P m.a.p. $\mathring{x}$.

Enligt kap. 4 är

$$\nabla_{\mathring{x}} P = F_x^T(x(t_1)) - 2 \cdot C \cdot (f - \mathring{x}) + \int_t^{t_1} \left[L_x^T + 2 \cdot C \cdot f_x^T (f - \mathring{x}) \right] ds$$

F_x, L_x är radvektorer med n (N) komponenter, medan f_x är en n.n (N.N) matris.

För en bestämd tidpunkt blir

$$\nabla_{\mathring{x}} P(t) = \begin{bmatrix} F_{x_1}(x(t_1)) - 2 \cdot C \cdot (f_1 - \mathring{x}_1) + \int_t^{t_1} \left[L_{x_1} + 2 \cdot C \cdot \sum_{i=1}^{n} f_{x_1}^{i} (f_i - \mathring{x}_i) \right] ds \\ \vdots \\ F_{x_n}(x(t_1)) - 2 \cdot C \cdot (f_n - \mathring{x}_n) + \int_t^{t_1} \left[L_{x_n} + 2 \cdot C \cdot \sum_{i=1}^{n} f_{x_n}^{i} (f_i - \mathring{x}_i) \right] ds \end{bmatrix}$$

Med utgångspunkt från denna vektor har nedanstående programavsnitt skrivits.

```
C      START COMPUTE GRADIENT XPNOM
C
  600 I=NSTP
      DO 595 L=1,N
  595 XVEC(L)=XNOM(L,I)
      CALL BBOUND
```

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

```
      DO 680 KP=1,N
      DO 640 I=1,NSTP
      TE=T0+(I-1)*H
      DO 610 L=1,N
610   XVEC(L)=XNOM(L,I)
      DO 620 K=1,NU
      N1=N+K
620   UVEC(K)=XSI(N1,I)
      CALL SYSTEM
      CALL PDER
      ETA(I)=0.
      DO 630 L=1,N
630   ETA(I)=ETA(I)+FX(L,KP)*(F(L)-XSI(L,I))
640   ETA(I)=2.*C*ETA(I)+SLX(KP)
      DO 650 I=1,NSTP
      J=NSTP-I+1
650   VETA(J)=ETA(I)
      CALL QSF(H,VETA,SINT,NSTP)
```

Kommentar: För varje komponent av gradienten beräknas uttrycket under integralen för alla tidpunkter och läggs i vektorn ETA.

P.g.a. att subrutinen QSF endat beräknar integraler av typen $\int_{t_o}^{t}$ där t varierar och vi vill beräkna integraler av typen $\int_{t}^{t_1}$ måste ETA vändas (läggs i VETA) .

Därefter anropas QSF varvid ETA integreras och resultatet läggs i SINT. Slutligen måste SINT vändas för att ge rätt resultat vid slutsummeringen (se nedan).

```
      DO 680 I=1,NSTP
      TE=T0+(I-1)*H
      DO 660 L=1,N
  660 XVEC(L)=XNOM(L,I)
      DO 670 K=1,NU
      N1=N+K
  670 UVEC(K)=XSI(N1,I)
      CALL SYSTEM
      CALL PDER
      J=NSTP-I+1
  680 GRADP2(KP,I)=SFX(KP)-2.*C*(F(KP)-XSI(KP,I))+SINT(J)
C     END GRADIENT XPNOM
```

10. Beskrivning av beräkning av P.

$$P(\mathring{x},u) = F(x(t_1)) + \int_{t_o}^{t_1} L(x,u)dt + C\cdot\langle f(x,u)-\dot{\mathring{x}}, f(x,u)-\dot{\mathring{x}}\rangle =$$

$$= F(x(t_1)) + \int_{t_o}^{t_1} \left[L(x,u) + C\cdot\left[(f(x,u)-\dot{\mathring{x}})^T(f(x,u)-\dot{\mathring{x}})\right]\right]dt$$

Beräkning av P görs i följande programavsnitt.

```
C      START COMPUTE THE VALUE OF THE FUNCTION P
C
       DO 728 I=1,NSTP
       TE=T0+(I-1)*H
       DO 722 L=1,N
  722  XVEC(L)=XNOM(L,I)
       DO 724 K=1,NU
       N1=K+N
  724  UVEC(K)=XSI(N1,I)
       CALL INTLOS
       CALL SYSTEM
       Q(I)=0.
       DO 726 L=1,N
  726  Q(I)=Q(I)+(F(L)-XSI(L,I))**2
  728  Q(I)=C*Q(I)+SL
       CALL QSF(H,Q,SINT,NSTP)
       I=NSTP
       DO 729 L=1,N
  729  XVEC(L)=XNOM(L,I)
       CALL FINLOS
       P=SINT(NSTP)+SF
C
C      END OF COMPUTATION OF P
```

Kommentar: 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 subrutinen USER vid sluttidpunkten och slutsummering sker.

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

Optimal lösning:

$$\hat{u}_1(t) = 6t-7$$
$$\hat{x}_1(t) = t^3 - \frac{7}{2}t^2 + 2t + 2$$
$$\hat{x}_2(t) = 3t^2 - 7t + 2$$

$$J_{min} = 26$$

För detta exempel kan man också beräkna hur minimivärdet av P beror av C: $P_{min} = 26 - \frac{28}{C}$ för

$$u^*(t) = 6t-7$$
$$x_1^*(t) = \left(\frac{1}{C}+1\right)\left(t^3 - \frac{7}{2}t^2\right) + 2t - \frac{6}{C}t + 2$$
$$x_2^*(t) = \left(\frac{1}{C}+1\right)\left(3t^2 - 7t + \frac{C}{C+1}\,2\right)$$

För att visa hur den numeriska noggrannheten beror på NSTEP (=antalet intervall som helatidsintervallet indelats i) anges nedan det P_{min}, som programmet beräknat för C=1 vid olika värden på NSTEP.

NSTEP	P_{min}
20	-76.73
100	-73.65
500	-72.32

Jfr. teoretiskt värde: $P_{min} = -72.0$

Dessa värden har tagits efter 7 iterationer (iteration = minimering längs en sökriktning) och värdet på SPROD2 var då 0.11808.

Ovanstående siffror visar tydligt 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.

Efter 27 iterationer erhölls då minimering gjorts för C=1,10,100 P_{min}= 24.86 (C=100).SPROD2 var då 0.14724 och SPROD3 = 0.00985.

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

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 datormaskin erhölls följande siffror då NSTEP =500:

		Totalt antal iterationer
C=1 EPS1=0.1	: 5 iterationer gav P_{min}=1.77	5
C=10 EPS1=0.1	: 4 iterationer gav P_{min}=1.977	9
C=100 EPS1=0.1	: 8 iterationer gav P_{min}=1.993	17
C=1000 EPS1=0.1	: 4 iterationer gav P_{min} =1.99551	21
C=10000 EPS1=0.1	: 5 iterationer gav P_{min} =1.99577	26

varvid SPROD3<0.00001.

Testexempel 3

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

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

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

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

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

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

$J_{min} = -2.6459$

Vid körning på datorskiva erhölls följande siffror då NSTEP=500

		Totalt antal iterationer
C=1 EPS1=0.1	: 11 iterationer gav $P_{min}=-2.998$	11
C=10 EPS1=0.1	: 9 iterationer gav $P_{min}=-2.684$	20
C=100 EPS1=0.1	: 2 iterationer gav $P_{min}=-2.6499$	22
C=1000 EPS1=0.1	: 3 iterationer gav $P_{min}=-2.64640$	25
C=10 000 EPS1=0.1	: 1 iteration gav $P_{min}=-2.64604$	26
C=100 000 EPS1=0.1	: 3 iterationer gav $P_{min}=-2.64600$	29

varvid SPROD3<0.00001

12. Jämförelseexempel.

System: $\dot{x}_1 = (1-x_2^2) x_1-x_2+u \quad x_1(0) = 0$

$\dot{x}_2 = x_1 \quad x_2(0) = 1$

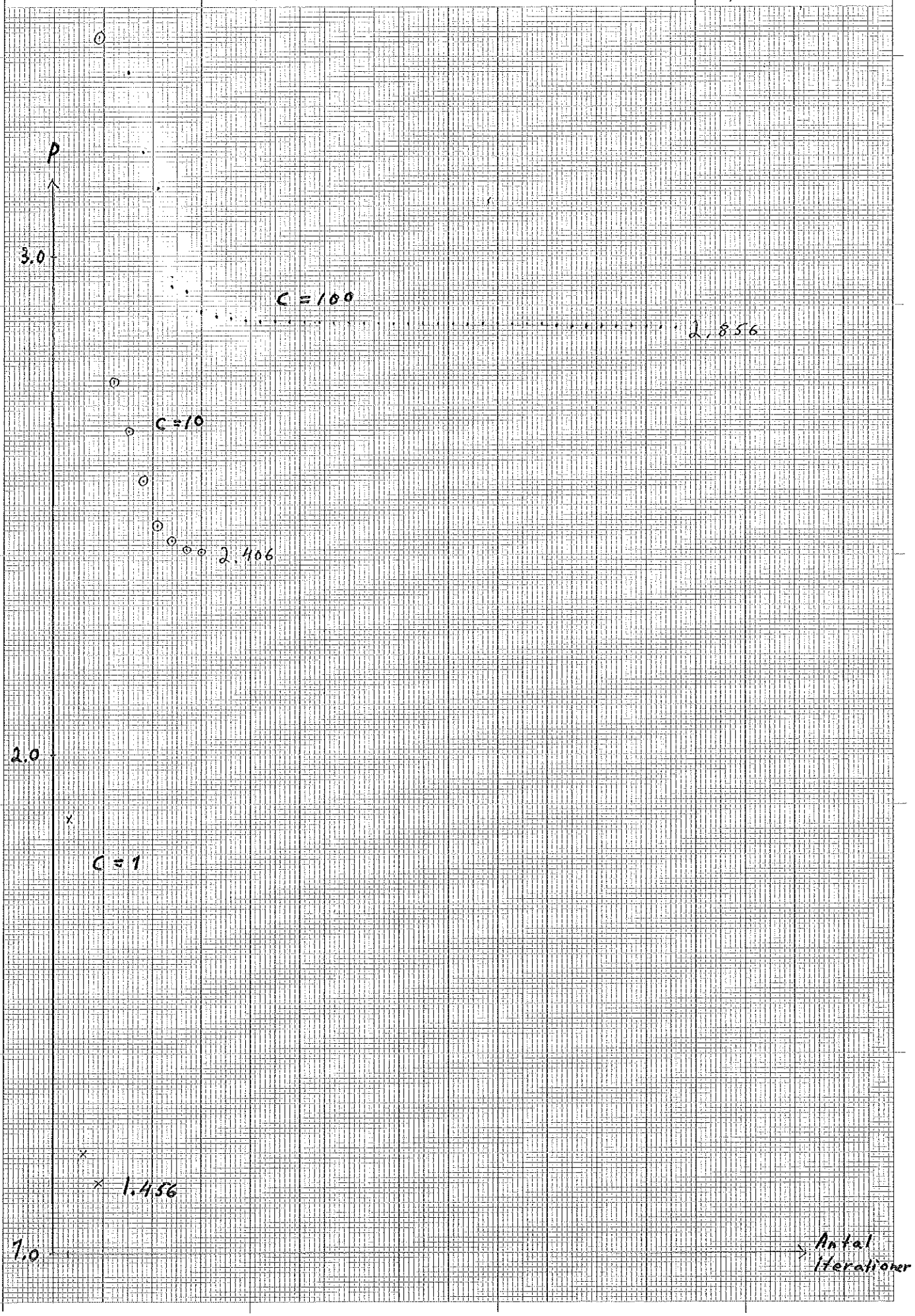
Förlustfunktion: $J = \int_0^5 (x_1^2+x_2^2+u^2) \, dt$

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 P som funktion av antalet iterationer för olika C-värden. Vid körning på dator erhölls efter 55 iterationer P min=2.856. 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.

Diagram 1

Jämförelse-exempel

P som funktion av antalet 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.

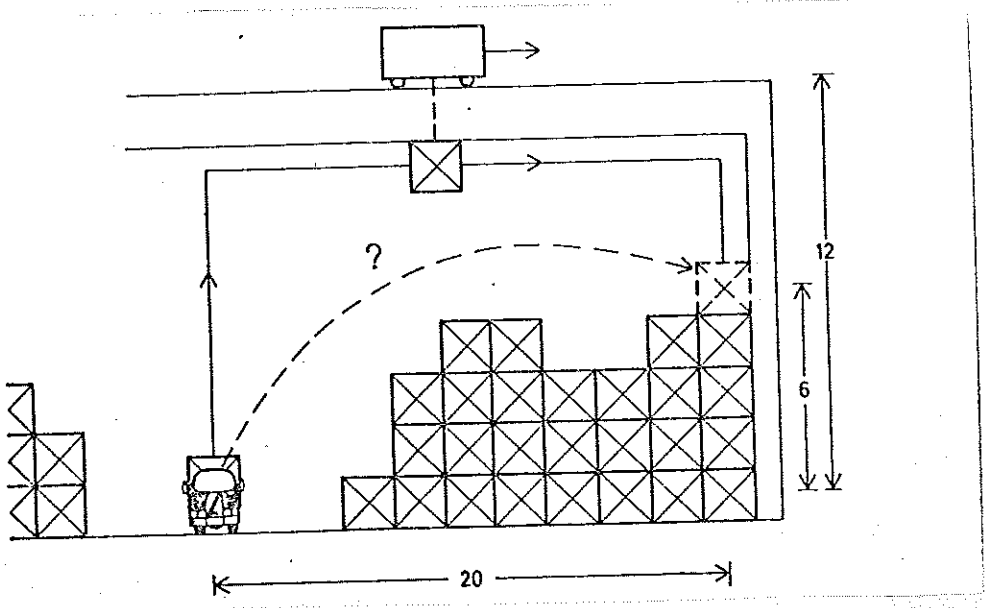
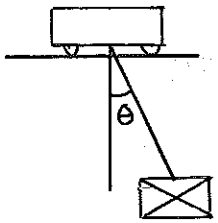


Fig. 1. Den speciella överföringen som minimeras. Alla avstånd i meter.

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

(x_1 = trallans läge, $\dot{x}_1$ = trallans hastighet, x_3 = θ, 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.

Diagram 2

Tillämpning-exempel

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

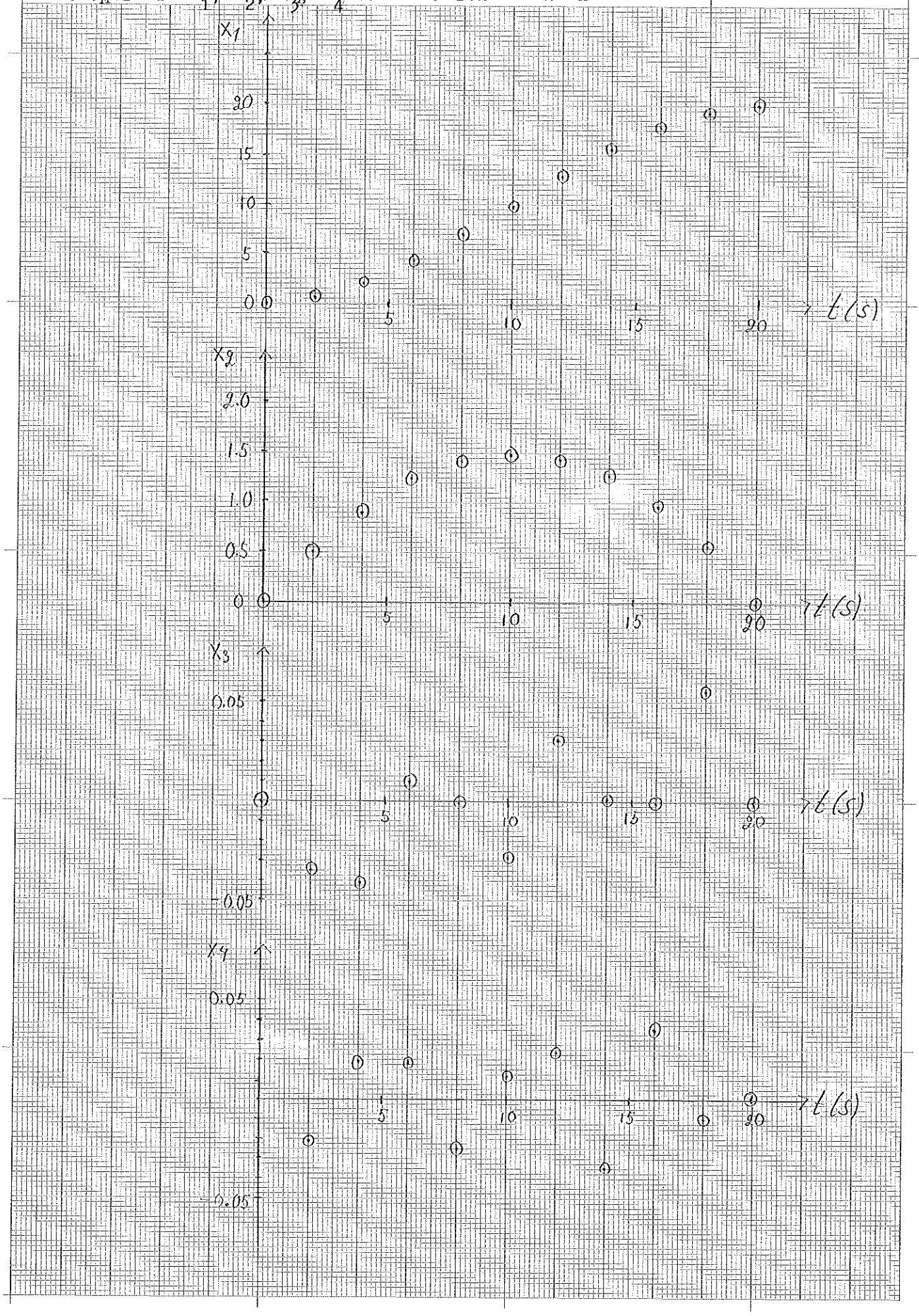
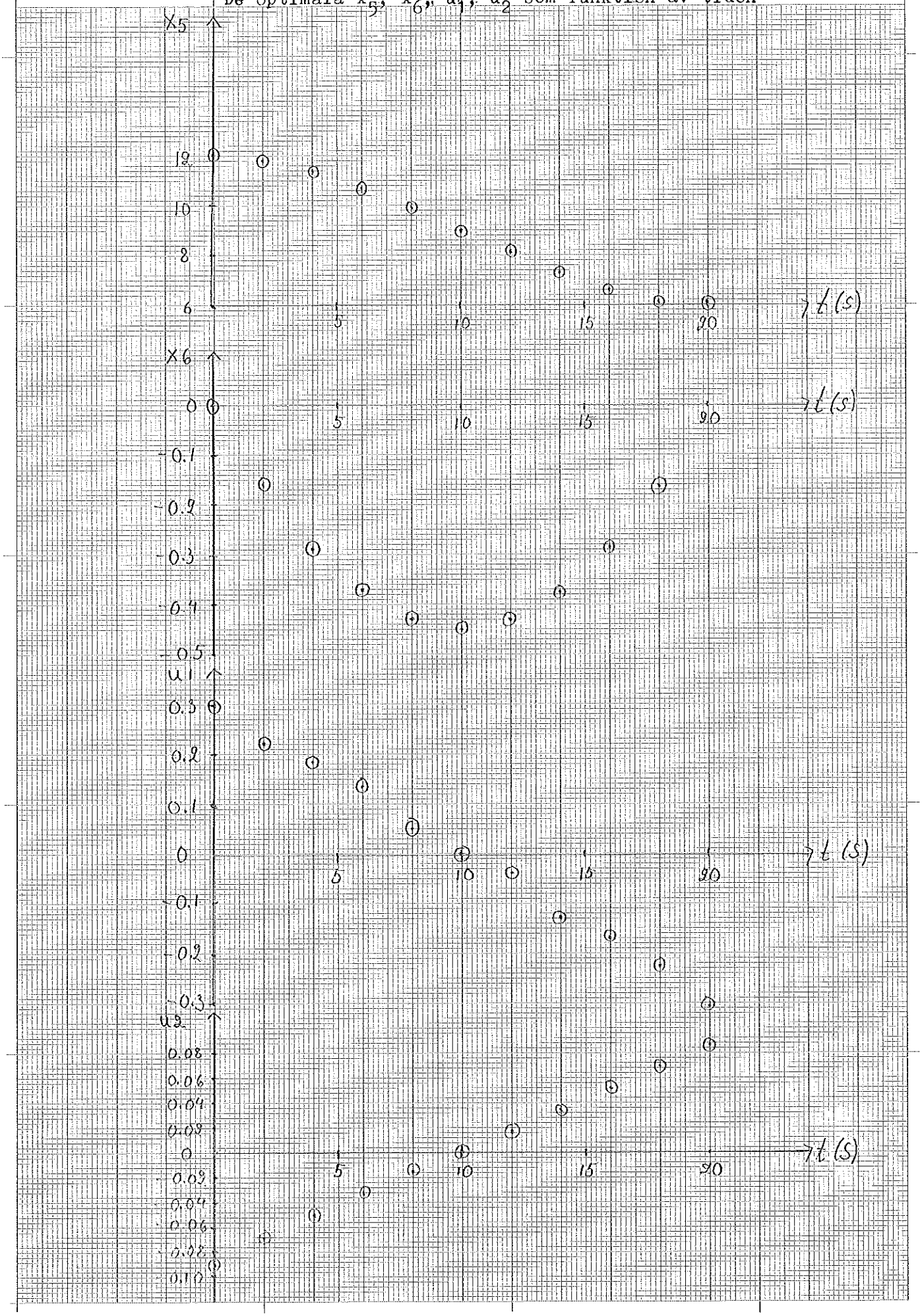


Diagram 3

Tillämpnings-
exempel

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



14. Egna erfarenheter och diskussion om eventuella förbättringar.

Riktvärden för valda parametrar:
NSTEP: Väljes till 500 så länge inte problemet är omfattande och kräver mycket räknearbete. I så fall får NSTEP minskas, vilket sker på bekostnad av den numeriska noggrannheten.

Initialgissning av XPNOM och UNOM: Sättes lika med noll vid enklare problem. I annat fall bör rimliga startvärden beräknas, om det är möjligt. Man kan ta de sista värdena från en tidigare körning om denna körning ej gav tillräckligt bra resultat.

EPS2: Väljes beroende på hur noggrannt man vill minimera, dvs. hur många C-värden man vill minimera för. Vanliga värden är mellan 0.01 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.

C: Man börjar att minimera för C=1, ökar sedan C till 10, 100 osv. Frågan är hur stort C man skall sluta minimera för. Det beror på hur stort exemplet är i fråga om räknearbete, konvergenshastighet och hur noga man vill minimera. För det mesta verkar det räcka med att minimera upp till C=100 eller C=1000.

EPS1: Då man i början inte vill få en noggrann minimering väljes EPS1 relativt stor i början för att sedan minskas för större C-värden. Värdena på EPS1 beror mycket på det exempel man kör, varför man ofta får prova sig fram. Som exempel på värden kan nämnas: (C=1) 25, (C=10) 10, (C=100) 1.

Eventuella förbättringar:
Det som man närmast kan tänka förbättra är den endimensionella minimeringen, som bör kunna göras effektivare.

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 mitt program och program [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.

Min egen metod: Se diagram 1. Samma typ av konvergens som för metod [2] . Konvergenshastigheten verkar vara ungefär som för metod [2] , men sämre än för metod [2] . 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 ett $C<C_0$ 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.

Metod [1] : Se diagram 1 i ref [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,R. ,"Straff-funktionsmetoder för numerisk lösning av optimala styrproblem". Examensarbete på Institutionen 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 L.S. Lasdoir,S.K. Mitter and A.D. Warren , "The conjugate gradient method for optimal control problems", IEEE Trans. Automatic Control,vol 12,1967,p 132-138.

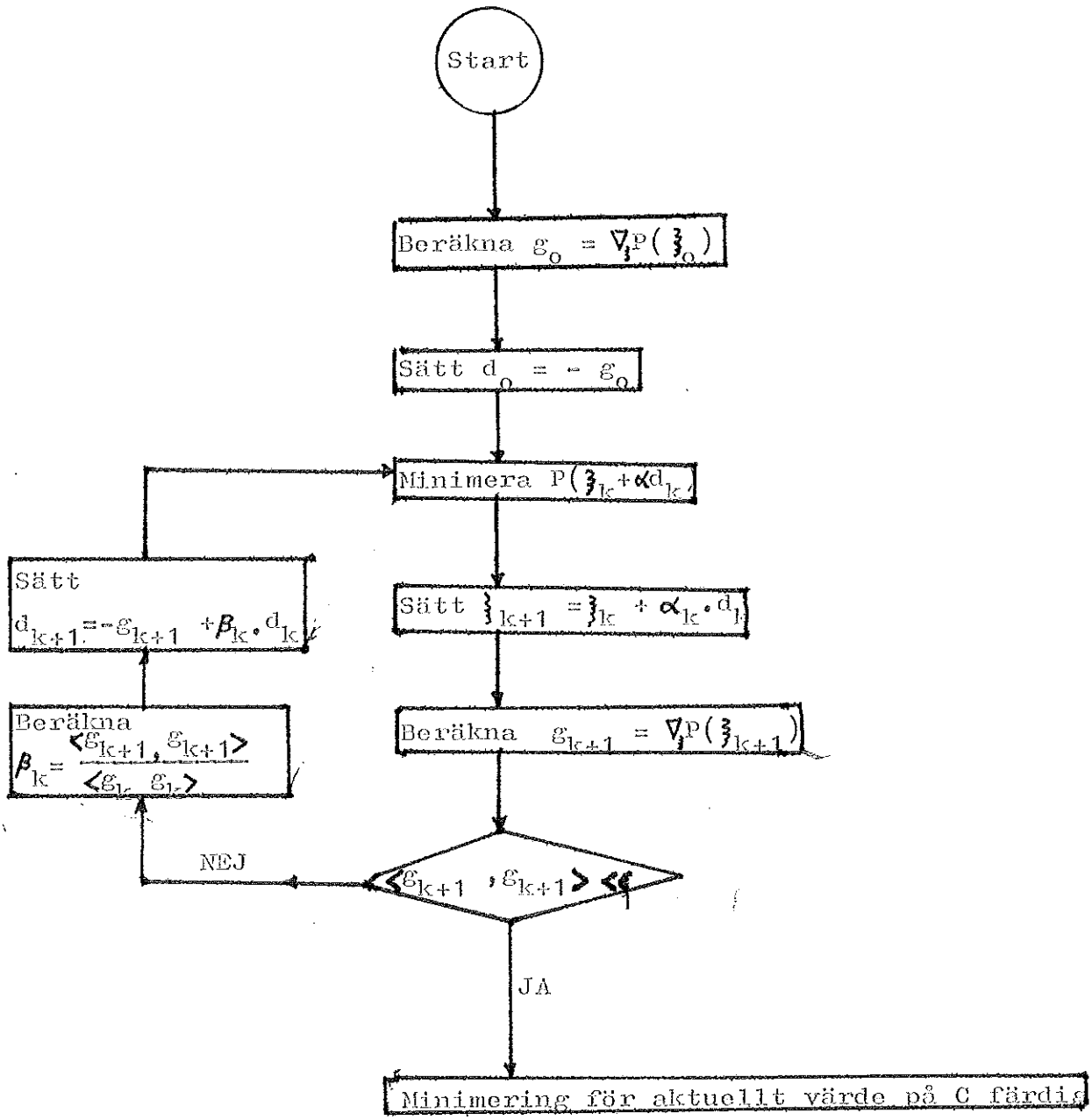
4 Luenberger,D.G. ,"Intoduction to linear and nonlinear programming", Addison Wesley,1973.

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 ξ_o.



Appendix B.

Exempel på USER.(se Jämförelseexemplet kap.12).

```
      SUBROUTINE USER
C
      DIMENSION XIN(6),UNOM(2,501),XPNOM(6,501),X(6),U(2),F(6),FX(6,6),F
     1U(6,2),SFX(6),SLX(6),SLU(2)
      COMMON/USER/ N,NU,NSTEP,H,T0,T1,NPRINT,XIN,XPNOM,UNOM,EPS1,EPS2,ST
     1EP,SF,SL,SFX,FX,FU,SLX,SLU,F,T,X,U,C
C
      ENTRY INIT
      N=2
      NU=1
      H=0.01
      NSTEP=500
      T0=0.
      T1=5.0
      NPRINT=100
      XIN(1)=0.
      XIN(2)=1.0
      XPNOM(1,1)=0.
      XPNOM(2,1)=0.
      UNOM(1,1)=0.
      DO 1 I=1,NSTEP
      XPNOM(1,I+1)=0.
      XPNOM(2,I+1)=0.
    1 UNOM(1,I+1)=0.
```

```
      EPS1=0.5
      EPS2=0.000000001
      STEP=0.1
      RETURN
C
      ENTRY CPARAM
      READ 1000,C
 1000 FORMAT(F10.1)
      RETURN
C
      ENTRY FINLOS
      SF=0.
      RETURN
C
      ENTRY INTLOS
      SL=X(1)**2+X(2)**2+U(1)**2
      RETURN
C
      ENTRY BROUND
      SFX(1)=0.
      SFX(2)=0.
      RETURN
C
      ENTRY PDER
      FX(1,1)=1.0-X(2)**2
      FX(1,2)=-2.0*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)=2.0*X(1)
      SLX(2)=2.0*X(2)
      SLU(1)=2.0*U(1)
      RETURN
C
      ENTRY SYSTEM
      F(1)=(1.0-X(2)**2)*X(1)-X(2)+U(1)
      F(2)=X(1)
      RETURN
      END
```

COMPILATION: NO DIAGNOSTICS.

Appendix C.

Listning av program IPF1.

```
C      PROGRAM IPF1
C
C
C
C
       DIMENSION XIN(6),UNOM(2,501),XPNOM(6,501),XNOM(6,501),XSI(8,501),Q
      1(501),SINT(501),GRADP1(8,501),GRADP2(8,501),RIKT(8,501),XVEC(6),UV
      1EC(2),ETA(501),VETA(501),SFX(6),F(6),FX(6,6),FU(6,2),SLX(6),SLU(2)
C
       COMMON/USER/ N,NU,NSTEP,H,T0,T1,NPRINT,XIN,XPNOM,UNOM,EPS1,EPS2,ST
      1EP,SF,SL,SFX,FX,FU,SLX,SLU,F,TE,XVEC,UVEC,C
C
C      INITIALIZE
       CALL INIT
       STEG=STEP
C
       PRINT 1000
 1000 FORMAT(32H1PRINTOUTS FROM THE PROGRAM IPF1,//)
       PRINT 1002,N,NU
 1002 FORMAT(18H NUMBER OF STATES=,I2,5X,28HNUMBER OF CONTROL VARIABLES=
      1,I2,/)
       PRINT 1004,T0,T1
 1004 FORMAT(14H INITIAL TIME=,F10.5,5X,11HFINAL TIME=,F10.5,/)
       PRINT 1006,NSTEP,H
 1006 FORMAT(40H THE TIME INTERVAL T0-T1 IS DIVIDED INTO,I5,36H INTERVAL
      1S (INTEGRATION STEP LENGTH=,F8.5,5X,1H),/)
       PRINT 1008,NPRINT
 1008 FORMAT(6H EVERY,I5,56HTH POINT OF THE TRAJECTORIES AND THE CONTROL
      1S IS PRINTED,/)
       PRINT 1010
 1010 FORMAT(35H THE INITIAL STATE OF THE SYSTEM IS,/)
       PRINT 1012,(XIN(I),I=1,N)
 1012 FORMAT(6F12.5)
       PRINT 1014
 1014 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 1016,T,(UNOM(J,I),J=1,NU)
 1016 FORMAT(F10.5,2(5X,F12.5))
   20  IP=IP+1
       PRINT 1018
 1018 FORMAT(/,29H THE GUESSED NOMINAL DX/DT IS,/,4X,4HTIME,10X,6H DX/DT
```

```
     1,/)
      IP=NPRINT
      DO 40 I=1,NSTP
      IF(IP-NPRINT)40,30,30
   30 T=T0+(I-1)*H
      IP=0
      PRINT 1020,T,(XPNOM(K,I),K=1,N)
 1020 FORMAT(F10.5,6(5X,F12.5))
   40 IP=IP+1
C
      NXSI=N+NU
C
C     INTEGRATE XPNOM TO GET XNOM
C
      DO 60 K=1,N
      DO 50 I=1,NSTP
   50 Q(I)=XPNOM(K,I)
      CALL QSF(H,Q,SINT,NSTP)
      DO 60 I=1,NSTP
   60 XNOM(K,I)=XIN(K)+SINT(I)
C
C     PRINT XNOM
C
      PRINT 1022
 1022 FORMAT(/,21H THE NOMINAL STATE IS,//,4X,4HTIME,10X,5H XNOM,/)
      IP=NPRINT
      DO 80 I=1,NSTP
      IF(IP-NPRINT)80,70,70
   70 T=T0+(I-1)*H
      IP=0
      PRINT 1024,T,(XNOM(K,I),K=1,N)
 1024 FORMAT(F10.5,6(5X,F12.5))
   80 IP=IP+1
C
C     PUT XPNOM AND UNOM INTO XSI
C
      DO 90 L=1,N
      DO 90 I=1,NSTP
   90 XSI(L,I)=XPNOM(L,I)
      DO 100 J=1,NU
      DO 100 I=1,NSTP
      K=J+N
  100 XSI(K,I)=UNOM(J,I)
C
C     START THE MAINLOOP
C
      STEP=STEG
  110 CALL CPARAM
      IF(C)120,130,130
  120 CALL EXIT
  130 PRINT 1026,C
 1026 FORMAT(/,25H THE CHOSEN VALUE OF C IS,F10.2,/)
      PRINT 1025,EPS1
 1025 FORMAT(/,28H THE CHOSEN VALUE OF EPS1 IS,F10.2,/)
C
C     COMPUTE THE GRADIENT AND THE VALUE OF THE FUNCTION P
C
      IND=1
      GO TO 600
  140 CONTINUE
C
C     PRINT THE VALUE OF THE FUNCTION P
C
      PRINT 1027,P
```

```
 1027 FORMAT(/,22H CHOSEN VALUES GIVE P=,F10.5,/)
C
C      PUT GRADP2 INTO GRADP1,CHOOSE SEARCHDIRECTION AND COMPUTE THE SCAL
C     1ARPRODUCT OF GRADP1 WITH ITSELF
C
       DO 150 I=1,NSTP
       Q(I)=0.
       DO 150 K=1,NXSI
       GRADP1(K,I)=GRADP2(K,I)
       RIKT(K,I)=-GRADP1(K,I)
  150  Q(I)=Q(I)+GRADP1(K,I)*GRADP1(K,I)
       CALL  QSF(H,Q,SINT,NSTP)
       SPROD1=SINT(NSTP)
       IF(SPROD1)154,154,156
  154  PRINT 1029
 1029  FORMAT(9H SPROD1=0)
       CALL EXIT
  156  CONTINUE
C
C      PRINT GRADP2
C
       PRINT 1028
 1028  FORMAT(/,16H THE GRADIENT IS,/,4X,4HTIME,10X,7H GRADP1,/)
       IP=NPRINT
       DO 170 I=1,NSTP
       IF(IP-NPRINT)170,160,160
  160  T=T0+(I-1)*H
       IP=0
       PRINT 1030,T,(GRADP1(K,I),K=1,NXSI)
 1030  FORMAT(F10.5,8(2X,F9.5))
  170  IP=IP+1
C
C      PRINT SPROD1
C
       PRINT 1032,SPROD1
 1032  FORMAT(/,8H SPROD1=,F10.5,/)
C
C      COUNTER FOR LOOP WHEN INTERPOLATION IS NOT SUCCESSFULL
C
       LCOUNT=0
C
C      COUNTER FOR THE MINIMISATION LOOP
C
       KCOUNT=0
C
C      COUNTER FOR HOW MANY TIMES THE VALUE OF THE FUNCTION P IS COMPUTED
C
       NFE=0
C
C
C      START OF THE MINIMISATION LOOP
C
  220  PB=P
C
C      COMPUTE THE MINIMISATION GRADIENT
C
       DO 230 I=1,NSTP
       Q(I)=0.
       DO 230 K=1,NXSI
  230  Q(I)=Q(I)+GRADP2(K,I)*RIKT(K,I)
       CALL QSF(H,Q,SINT,NSTP)
       GB=SINT(NSTP)
C
C      CHECK IF THE MINIMISATION GRADIENT IS NEGATIVE
```

```
C
      IF(GB)260,260,240
C
C     MINIMISATION GRADIENT POSITIVE NEW DIRECTION RIKT=-RIKT
C
  240 DO 250 I=1,NSTP
      DO 250 K=1,NXSI
  250 RIKT(K,I)=-RIKT(K,I)
      GB=-GB
      PRINT 1036
 1036 FORMAT(/,11H RIKT=-RIKT,/)
  260 ALFA=STEP
C
C     COUNTER FOR EXTRAPOLATION LOOP
      MCOUNT=0
C
C     START EXTRAPOLATION LOOP
C
  280 PA=PB
      GA=GB
      MCOUNT=MCOUNT+1
C
C     COMPUTE NEW XSI
C
      DO 300 I=1,NSTP
      DO 300 K=1,NXSI
  300 XSI(K,I)=XSI(K,I)+ALFA*RIKT(K,I)
C
C     INTEGRATE XPNOM TO GET XNOM
C
      DO 320 K=1,N
      DO 310 I=1,NSTP
  310 Q(I)=XSI(K,I)
      CALL QSF(H,Q,SINT,NSTP)
      DO 320 I=1,NSTP
  320 XNOM(K,I)=SINT(I)+XIN(K)
C
C     COMPUTE THE GRADIENT AND THE VALUE OF THE FUNCTION P
C
      IND=2
      GO TO 600
  370 CONTINUE
C
C     PRINT P
C
      PRINT 1038,P
 1038 FORMAT(/,22H EXTRAPOLATION GIVE P=,F10.5,/)
      PB=P
      NFE=NFE+1
C
C     COMPUTE THE MINIMISATION GRADIENT GB
C
      DO 380 I=1,NSTP
      Q(I)=0.
      DO 380 K=1,NXSI
  380 Q(I)=Q(I)+GRADP2(K,I)*RIKT(K,1)
      CALL QSF(H,Q,SINT,NSTP)
      GB=SINT(NSTP)
C
C     EXTRAPOLATION OR INTERPOLATION?
C
      IF((GB.GE.0).OR.(PB.GE.PA))GO TO 400
C
C     EXTRAPOLATION
```

```
C
  390 ALFA=4*ALFA
      STEP=4*STEP
      IF(MCOUNT.LT.20)GO TO 280
C
C     NOT SUCCESSFULL EXTRAPOLATION
C
      PRINT 1040
 1040 FORMAT(/,23H MCOUNT GREATER THAN 20,/)
      CALL EXIT
C
C     START INTERPOLATION
C
C     PRINT MCOUNT
C
  400 PRINT 1042,MCOUNT
 1042 FORMAT(/,8H MCOUNT=,I2,/)
      NCOUNT=0
C
C     COUNTER FOR INTERPOLATION LOOP
C
  410 NCOUNT=NCOUNT+1
C
C     CUBIC INTERPOLATION
C
      Z=3.*(PA-PB)/ALFA+GA+GB
      S=Z**2-GA*GB
      IF(S.LT.0.)S=0.
      W=SQRT(S)
      SLAMDA=ALFA*(GB+W-Z)/(GB-GA+2.0*W)
C
C     COMPUTE NEW XSI
C
      DO 420 I=1,NSTP
      DO 420 K=1,NXSI
  420 XSI(K,I)=XSI(K,I)-SLAMDA*RIKT(K,I)
C
C     INTEGRATE XPNOM TO GET XNOM
C
      DO 440 K=1,N
      DO 430 I=1,NSTP
  430 Q(I)=XSI(K,I)
      CALL QSF(H,Q,SINT,NSTP)
      DO 440 I=1,NSTP
  440 XNOM(K,I)=SINT(I)+XIN(K)
C
C     COMPUTE THE GRADIENT AND THE VALUE OF THE FUNCTION P
C
      IND=3
      GO TO 600
  490 CONTINUE
C
C     PRINT P
C
      PRINT 1044,P
 1044 FORMAT(/,22H INTERPOLATION GIVE P=,F10.5,/)
C
      NFE=NFE+1
C
C     CHECK IF THE FUNCTION VALUE IN THE INTERPOLATED POINT IS LESS COMP
C    1ARED TO THE FUNCTION VALUES OF THE OLD POINTS
C
      IF(P.GT.PA.OR.P.GT.PB)GO TO 510
C
```

```
C      CHOOSE THE INTERPOLATED POINT AS THE MINIMIZING POINT ALONG THE SE
C     1ARCHDIRECTION
C
       PRINT 1144
 1144 FORMAT(/,85H CHOOSE THE INTERPOLATED POINT AS THE MINIMIZING POINT
     1 ALONG THE SEARCHDIRECTION RIKT,/)
  500 GO TO 730
  510 STEP=STEP/4
C
C      CHECK IF THE FUNCTION VALUE OF THE RIGHT POINT IS LESS THAN THE VA
C     1LUE OF THE LEFT ONE
C
       IF(PB.GE.PA)GO TO 560
C
C      CHOOSE THE RIGHT POINT AS THE MINIMIZING POINT ALONG THE SEARCHDIR
C     1ECTION
C
       PRINT 1145
 1145 FORMAT(/,73H CHOOSE THE RIGHT POINT AS THE MINIMIZING POINT ALONG
     1THE SEARCHDIRECTION ,/)
  520 DO 530 I=1,NSTP
       DO 530 K=1,NXSI
  530 XSI(K,I)=XSI(K,I)+SLAMDA*RIKT(K,I)
C
C      INTEGRATE XPNOM TO GET XNOM
C
       DO 550 K=1,N
       DO 540 I=1,NSTP
  540 Q(I)=XSI(K,I)
       CALL QSF(H,Q,SINT,NSTP)
       DO 550 I=1,NSTP
  550 XNOM(K,I)=SINT(I)+XIN(K)
       GO TO 600
C
C      CHOOSE THE INTERPOLATED POINT AS THE RIGHT POINT
C      COMPUTE THE MINIMISATION GRADIENT
C
  560 DO 570 I=1,NSTP
       Q(I)=0.
       DO 570 K=1,NXSI
  570 Q(I)=Q(I)+GRADP2(K,I)*RIKT(K,I)
       CALL QSF(H,Q,SINT,NSTP)
       GB=SINT(NSTP)
C
       PB=P
       ALFA=ALFA-SLAMDA
       IF(NCOUNT.LT.20)GO TO 410
C
C      NOT SUCCESSFULL INTERPOLATION.CHOOSE THE LAST INTERPOLATED POINT A
C     1S THE MINIMIZING POINT.CHOOSE SEARCHDIRECTION AS THE NEGATIVE GRAD
C     1IENT
C
  580 LCOUNT=LCOUNT+1
       DO 590 I=1,NSTP
       DO 590 K=1,NXSI
  590 RIKT(K,I)=-GRADP2(K,I)
       STEP=STEG
       PRINT 1045
 1045 FORMAT(/,23H NCOUNT GREATER THAN 20,/)
       IF(LCOUNT.LT.5)GO TO 592
       PRINT 1047
 1047 FORMAT(/,22H LCOUNT GREATER THAN 5,/)
       CALL EXIT
  592 GO TO 220
```

```
C
C       START COMPUTE THE GRADIENT.THE GRADIENT IS PUT INTO GRADP2
C       START COMPUTE GRADIENT XPNOM
C
  600 I=NSTP
      DO 595 L=1,N
  595 XVEC(L)=XNOM(L,I)
      CALL BBOUND
      DO 680 KP=1,N
      DO 640 I=1,NSTP
      TE=T0+(I-1)*H
      DO 610 L=1,N
  610 XVEC(L)=XNOM(L,I)
      DO 620 K=1,NU
      N1=N+K
  620 UVEC(K)=XSI(N1,I)
      CALL SYSTEM
      CALL PDER
      ETA(I)=0.
      DO 630 L=1,N
  630 ETA(I)=ETA(I)+FX(L,KP)*(F(L)-XSI(L,I))
  640 ETA(I)=2.*C*ETA(I)+SLX(KP)
      DO 650 I=1,NSTP
      J=NSTP-I+1
  650 VETA(J)=ETA(I)
      CALL QSF(H,VETA,SINT,NSTP)
      DO 680 I=1,NSTP
      TE=T0+(I-1)*H
      DO 660 L=1,N
  660 XVEC(L)=XNOM(L,I)
      DO 670 K=1,NU
      N1=N+K
  670 UVEC(K)=XSI(N1,I)
      CALL SYSTEM
      CALL PDER
      J=NSTP-I+1
  680 GRADP2(KP,I)=SFX(KP)-2.*C*(F(KP)-XSI(KP,I))+SINT(J)
C       END GRADIENT XPNOM
C
C       START COMPUTE GRADIENT UNOM
      DO 720 I=1,NSTP
      TE=T0+(I-1)*H
      DO 690 L=1,N
  690 XVEC(L)=XNOM(L,I)
      DO 700 K=1,NU
      N1=N+K
  700 UVEC(K)=XSI(N1,I)
      CALL PDER
      CALL SYSTEM
      DO 720 J=1,NU
      SUM=0.
      DO 710 L=1,N
  710 SUM=SUM+FU(L,J)*(F(L)-XSI(L,I))
      K=J+N
  720 GRADP2(K,1)=SLU(J)+2.*C*SUM
C
C       END GRADIENT UNOM
C
C       END GRADIENT
C
C       START COMPUTE THE VALUE OF THE FUNCTION P
C
      DO 728 I=1,NSTP
      TE=T0+(I-1)*H
```

```
      DO 722 L=1,N
  722 XVEC(L)=XNOM(L,I)
      DO 724 K=1,NU
      N1=K+N
  724 UVEC(K)=XSI(N1,I)
      CALL INTLOS
      CALL SYSTEM
      Q(I)=0.
      DO 726 L=1,N
  726 Q(I)=Q(I)+(F(L)-XSI(L,1))**2
  728 Q(I)=C*Q(I)+SL
      CALL QSF(H,Q,SINT,NSTP)
      I=NSTP
      DO 729 L=1,N
  729 XVEC(L)=XNOM(L,I)
      CALL FINLOS
      P=SINT(NSTP)+SF
C
C     END OF COMPUTATION OF P
C
C     JUMP BACKWARDS WHEN USING THE COMPUTATION OF THE GRADIENT AND P
C
      IF(IND)735,730,735
  735 JUMP=IND
      IND=0
      GO TO(140,370,490),JUMP
C
CC     PRINT NFE,LCOUNT,NCOUNT
C
  730 PRINT 1046,NFE,LCOUNT,NCOUNT
 1046 FORMAT(/,65H VALUES WHEN OUT OF THE MINIMISATION LOOP FOR THE SEAR
     1CHDIRECTION,/,5H NFE=,I3,8H LCOUNT=,I3,8H NCOUNT=,I3,/)
C
C     PRINT XSI
C
      PRINT 1048
 1048 FORMAT(/,81H THE VALUES OF XSI WHEN OUT OF THE MINIMISATION LOOP F
     1OR THE SEARCHDIRECTION RIKT,/,4X,4HTIME,10X,4H XSI,/)
      IP=NPRINT
      DO 750 I=1,NSTP
      IF(IP-NPRINT)750,740,740
  740 T=T0+(I-1)*H
      IP=0
      PRINT 1050,T,(XSI(K,I),K=1,NXSI)
 1050 FORMAT(F10.5,8(2X,F9.5))
  750 IP=IP+1
C
C     PRINT XNOM
C
      PRINT 1052
 1052 FORMAT(/,81H THE VALUE OF XNOM WHEN OUT OF THE MINIMISATION LOOP F
     1OR THE SEARCHDIRECTION RIKT,/,4X,4HTIME,10X,5H XNOM,/)
      IP=NPRINT
      DO 770 I=1,NSTP
      IF(IP-NPRINT)770,760,760
  760 T=T0+(I-1)*H
      IP=0
      PRINT 1054,T,(XNOM(K,I),K=1,N)
 1054 FORMAT(F10.5,6(5X,F12.5))
  770 IP=IP+1
C
C     PRINT P
C
      PRINT 1056,P
```

```
 1056 FORMAT(/,94H THE VALUE OF THE FUNCTION P WHEN OUT OF THE MINIMISAT
     1ION LOOP FOR THE SEARCHDIRECTION RIKT P=,F10.5,/)
C
C      COMPUTE  THE SCALARPRODUCT OF GRADP2 WITH  ITSELF=  SPROD2
C
       DO 780 I=1,NSTP
       Q(I)=0.
       DO 780 K=1,NXSI
  780  Q(I)=Q(I)+GRADP2(K,I)*GRADP2(K,I)
       CALL QSF(H,Q,SINT,NSTP)
       SPROD2=SINT(NSTP)
C
C      PRINT SPROD2
C
       PRINT 1058,SPROD2
 1058 FORMAT(/,8H SPROD2=,F10.5,/)
C
C      CHECK IF THE MINIMISATION FOR A GIVEN C IS FINISHED
C
       IF(SPROD2-EPS1)830,830,790
C
C      MINIMISATION FOR A GIVEN C NOT FINISHED
C
  790  KCOUNT=KCOUNT+1
       BETA=SPROD2/SPROD1
       PRINT 1060,BETA
 1060 FORMAT(/,6H BETA=,F10.5)
       SPROD1=SPROD2
C
C      COMPUTE NEW SEARCHDIRECTION RIKT
C      PUT GRADP2 INTO GRADP1
       DO 800 K=1,NXSI
       DO 800 I=1,NSTP
       RIKT(K,I)=-GRADP2(K,I)+BETA*RIKT(K,I)
  800  GRADP1(K,1)=GRADP2(K,I)
C
C      PRINT RIKT
C
       PRINT 1061
 1061 FORMAT(/,50H THE VALUES OF THE ACTUAL SEARCHDIRECTION RIKT ARE,/,4
     1X,4HTIME,10X,5H RIKT,/)
       IP=NPRINT
       DO 820 I=1,NSTP
       IF(IP-NPRINT)820,810,810
  810  T=T0+(I-1)*H
       IP=0
       PRINT 1062,T,(RIKT(K,I),K=1,NXSI)
 1062 FORMAT(F10.5,8(2X,F9.5))
  820  IP=IP+1
       GO TO 220
C
C      MINIMISATION FOR A GIVEN C IS FINISHED
C      COMPUTE THE SCALARPRODUCT OF  F-XPNOM WITH ITSELF = SPROD3
C
  830  DO 860 I=1,NSTP
       DO 840 L=1,N
  840  XVEC(L)=XNOM(L,I)
       DO 850 K=1,NU
       N1=N+K
  850  UVEC(K)=XSI(N1,I)
       TE=T0+(I-1)*H
       CALL SYSTEM
       Q(I)=0.
       DO  860 L=1,N
```

```
  860 Q(I)=Q(I)+(F(L)-XSI(L,1))**2
      CALL QSF(H,Q,SINT,NSTP)
      SPROD3=SINT(NSTP)
C
C      PRINT KCOUNT,SPROD3
C
      PRINT 1064,KCOUNT,SPROD3
 1064 FORMAT(/,60H VALUES WHEN OUT OF THE MINIMISATION LOOP FOR ALL DIRE
     1CTIONS,/,8H KCOUNT=,I3,8H SPROD3=,F10.5,/)
C
C     CHECK IF THE EQUATION DX/DT=F IS SATISFIED
C
      IF(SPROD3-EPS2)870,870,110
C
C     PRINT THE OPTIMAL VALUES OF XNOM AND UNOM
  870 PRINT 1066
 1066 FORMAT(/,34H THIS IS THE OPTIMAL VALUE OF XNOM,/)
      IP=NPRINT
      DO 890 I=1,NSTP
      IF(IP-NPRINT)890,880,880
  880 T=T0+(I-1)*H
      IP=0
      PRINT 1068,T,(XNOM(K,I),K=1,N)
 1068 FORMAT(F10.5,6(5X,F9.5))
  890 IP=IP+1
      PRINT 1070
 1070 FORMAT(/,34H THIS IS THE OPTIMAL VALUE OF UNOM,/)
      IP=NPRINT
      DO 920 I=1,NSTP
      IF(IP-NPRINT)920,900,900
  900 T=T0+(I-1)*H
      IP=0
      DO 910 K=1,NU
      N1=K+N
  910 UNOM(K,I)=XSI(N1,1)
      PRINT 1072,T,(UNOM(K,I),K=1,NU)
 1072 FORMAT(F10.5,2(5X,F9.5))
  920 IP=IP+1
      CALL EXIT
      END

OMPILATION:        NO  DIAGNOSTICS.
```