IDENTIFIERING AV OLINJÄRA SYSTEM

KERSTIN BÅÅTH

Rapport R E - 15 sept. 1967

IDENTIFERING AV OLINJÄRA SYSTEM.

Kerstin Bååth

INNEHÅLL

I

INLEDNING

Idetta examensarbete diskuteras en metod som kan användas då en process beskrives med en olineär differentialekvation med känd struktur men med ett antal okända koefficienter. Dessa kan t.ex. vara reaktionshastigheter i en metallurgisk process. En metod för bestämning av dessa okända koefficienter beskrives. Proceduren har testats på artificiellt genererade data.

Först studeras metoden i det enklare fallet med endast en olineär ekvation, varefter den även begagnas i det allmänna fallet. ALGOL-program för de båda fallen konstrueras även. Idet allmänna fallet är programmet uppdelat i en generell del och en del specifik för det system vi studerar. Den senare delen måste omprogrammeras för varje nytt fall.

II

PROBLEMSTÄLLNING

Betrakta ett system som beskrives av diffrentialekvationerna

$$\frac{dx}{dt} = f(x,u,a)$$

$$y = g(x,u,a)$$

Här är

x tillståndsvariabeln

u insignalen

y utsignalen

Det förutsättes att funktionerna $f(x,u,a)$ och $g(x,u,a)$ är kända i x, u och a men parametern a är okänd. Avsikten är att bestämma denna okända parameter. Vi förfar på följande sätt. En godtycklig insignal $u(t)$ väljes och därefter observeras utsignalen $y(t)$. För enkelhets skull arbetar vi med det diskreta systemet i stället för det tidskontinuerliga. D.v.s.

$$x(t+T) = g(x(t),u(t),a)$$

Där $T=1$, $t=1,2,3\ldots$

I det följande antages att $x(t)$ kan mätas direkt. Mätvärdet kallas $y(t)$.

III.1

LÖSNING, ENKLARE FALLET

Betrakta först det skalära fallet. Ett sätt att bestämma den okända parametern a är att använda minstakvadratmetoden (ref.1).

Vi inför förlustfunktionen

$$V=\sum_{t=1}^{N}(y(t)-x(t))^2$$

Där y(t) är mätningen av x vid tidpunkten t.

$$x(t+1)=g(x(t),u(t),a) \qquad t=0,1,2...$$

Även x(0) är alltså en parameter.

Vi skall bestämma parametern a på ett sådant sätt att förlustfunktionen blir så liten som möjligt.

$$V= V(a,x(0))$$

För lösning av minimeringsproblemet beräknas derivatan av V med avseende på a. Därefter väljes a så att derivatan blir noll.

$$\frac{dv}{da}=0$$

löses med Newton-Raphsons metodik(ref.1).

Det x som löser f(x) =0 kan erhållas iterativt ur

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)} \qquad n=0,1,2...$$

om startvärdet x_0 valts så att konvergens erhålles (ref.3).

Vi betraktar åter

$$x(t+1) = g(x(t),u(t),a) \qquad t=0,1,2...$$

Som förlustfunktion väljes

$$V = \frac{1}{2}\sum_{t=1}^{N}(y(t)-x(t))^2$$

Vi antar att x(0) är känd d.v.s. V = V(a)

Minimering enligt följande

$$\frac{dV}{da} = -\sum_{t=1}^{N}(y(t) - x(t))\frac{dx(t)}{da}$$

$$\frac{d^2V}{da^2} = -\sum_{t=1}^{N}(y(t) - x(t))\frac{d^2x(t)}{da^2} + \frac{dx(t)}{da} \cdot \frac{dx(t)}{da} \qquad \text{I}$$

Här är

$$\frac{dx(t+1)}{da} = \frac{dg}{dx}(x(t),u(t),a)\frac{dx(t)}{da} + \frac{dg}{da}(x(t),u(t),a)$$

$$\frac{d^2x(t+1)}{da^2} = \frac{d^2g}{dx^2}(x(t),u(t),a) \cdot \left(\frac{dx(t)}{da}\right)^2 + \frac{d^2g}{dadx}(x(t),u(t),a) \cdot \frac{dx(t)}{da} +$$

$$\frac{dg}{dx}(x(t),u(t),a)\frac{d^2x(t)}{da^2} + \frac{d^2g}{dxda}(x(t),u(t),a)\frac{dx(t)}{da} + \frac{d^2g}{da^2}(x(t),u(t),a)$$

Det visar sig att första termen i ekv. I är jämförelsevis liten och vi försummar därför i fortsättningen denna.

Vid körning av problem på datamaskin är det lämpligt att i så hög grad som möjligt använda rekursiva formler.På det sättet begränsar man det antal celler som behövs för lagring av data.

Införes rekursiva formler erhålles

$$V(t+1) = \frac{1}{2}\sum_{t=1}^{t+1}(y(t)-x(t))^2 = V(t) + \frac{1}{2}(y(t+1) - x(t+1))^2$$

$$\frac{dV(t+1)}{da} = \frac{dV(t)}{da} - (y(t+1) - x(t+1))\frac{dx(t+1)}{da}$$

$$\frac{d^2V(t+1)}{da^2} \approx \frac{d^2V(t)}{da^2} + \left(\frac{dx(t+1)}{da}\right)^2$$

Med N-R:s metod löses $\frac{dV}{da} = 0$. Då erhålles

$$a_{k+1} = a_k - \frac{d^2V^{-1}}{da^2} \cdot \frac{dV}{da} \qquad k = 0,1,2....$$

Där a_0 är lämpligt vald för att konvergens skall erhållas.

III.2

EXEMPEL. GENERERING AV DATA

Den ovan skisserade lösningsmetodiken skall studeras i ett enkelt fall.

Problem:

$$\frac{dx}{dt} = -ax^3$$

Här är x tillståndsvariabeln

a okänd parameter

En differensekvation kan erhållas genom direkt lösning av ovanstående ekvation enligt följande.

$$-\frac{dx}{x^3} = a\,dt$$

$$\frac{1}{2x^2(t+T)} - \frac{1}{2x^2(t)} = aT$$

$$x(t+T) = \frac{x(t)}{(1+2aTx^2(t))^{\frac{1}{2}}}$$

Välj $a = 1$

$T = 1$

$x(0) = 1$

Då fås

$x(0) = 1$

$x(1) = 0,5774$

$x(2) = 0,4472$

$x(3) = 0,3780$

$x(4) = 0,3333$

$x(5) = 0,3015$

$x(6) = 0,2773$

$x(7) = 0,2582$

$x(8) = 0,2425$

$x(9) = 0,2294$

$x(10) = 0,2182$

Vi antar nu att a är okänd. Istället har vi tillgång till ovanstående data som mätvärden y och vill härur bestämma värdet på a. Gissa ett a t.ex. a = 0,5

Begynnelsevärdet x(0) = 1 anses känt.

Användes

$$x(t+T) = g(x(t),a) = \left(\frac{x(t)}{1+2ax^2(t)}\right)^{\frac{1}{2}}$$ erhålles

$$\frac{dg}{dx} = \left(\frac{1}{1+2ax^2(t)}\right)^{3/2}$$

$$\frac{dg}{da} = \left(\frac{-x^3(t)}{1+2ax^2(t)}\right)^{3/2}$$

Med andvändning av de rekursiva formler som utvecklats i III.1 fås vid räkning på bordsmaskin följande. För enkelhets skull har endast hälften av y-värdena använts.

TABELL

t	y	x	$\frac{dg}{dx}$	$\frac{dg}{da}$	$\frac{dx}{da}$	V	$\frac{dV}{da}$	$\frac{d^2V}{da^2}$
0	1	1	0,3535	-0,3535	0	0	0	0
1	0,5774	0,7071	0,5444	-0,1925	-0,3535	0,0084	-0,0458	0,1250
2	0,4472	0,5774	0,6494	-0,1250	-0,3850	0,0169	-0,0959	0,2732
3	0,3780	0,5000	0,7156	-0,0895	-0,3750	0,0243	-0,1417	0,4138
4	0,3333	0,4472	0,7607	-0,0604	-0,3779	0,0308	-0,1847	0,5566
5	0,3015	0,4082	0,7938	-0,0540	-0,3478	0,0365	-0,2218	0,6776
0	1	1	0,2313	-0,2313	0	0	0	0
1	0,5774	0,6138	0,4836	-0,1118	-0,2313	0,0007	-0,0084	0,0535
2	0,4472	0,4818	0,6142	-0,0687	-0,2237	0,0013	-0,0161	0,1035
3	0,3780	0,4096	0,6925	-0,0476	-0,2061	0,0018	-0,0226	0,1460
4	0,3333	0,3624	0,7442	-0,0354	-0,1903	0,0022	-0,0281	0,1822
5	0,3015	0,3285	0,7816	-0,0277	-0,1771	0,0026	-0,0329	0,2136
0	1	1	0,2397	-0,2397	0	0	0	0
1	0,5774	0,5810	0,4665	-0,0918	-0,2397	$6 \cdot 10^{-6}$	-0,0009	0,0575
2	0,4472	0,4506	0,6048	-0,0553	-0,2036	$12 \cdot 10^{-6}$	-0,0016	0,0990
3	0,3780	0,3811	0,6865	-0,0380	-0,1784	$17 \cdot 10^{-6}$	-0,0022	0,1308
4	0,3333	0,3362	0,7405	-0,0281	-0,1605	$21 \cdot 10^{-6}$	-0,0027	0,1566
5	0,3015	0,3042	0,7786	-0,0219	-0,1470	$23 \cdot 10^{-6}$	-0,0030	0,1782

Med startvärdet $a_0 = 0,5$ erhölls

$$a_1^1 = 0,877$$

$$a_2 = 0,9810$$

$$a_3 = 0,99784$$

Det är sålunda god konvergens mot det av oss kända värdet $a = 1$. Metoden fungerade och vi skall nu även formulera ett ALGOL-program för beräkningarna.

III.3

PROGRAMMERING

Ett program för lösning av det tidigare behandlade problemet skall skrivas i ALGOL för körning på Siffermaskinen i Lund, SMIL. Stansning av programmet sker på åttakanalsremsa enligt konventionerna i SMIL-ALGOL (ref.2).Indata stansas på femkanalsremsa. Utskrift erhålles normalt på radskrivare men i något fall begäres utskrift på femkanals hålremsa.

Program 1:

I början av programmet deklareras fyra funktionsprocedurer. De är med beteckningar enligt det föregående

$$g = g(x,a)$$

$$gx = \frac{dg}{dx}(x,a)$$

$$ga = \frac{dg}{da}(x,a)$$

samt Newton som innehåller beräkningen av nytt a-värde enligt N-R:s metod.

De deklarerade variablerna är

a	okänd parameter
x	tillståndsvariabel
dx	$\frac{dx}{da}$
v	förlustfunktionen
dv	$\frac{dv}{da}$
ddv	$\frac{d^2v}{da^2}$
h,r	hjälpvariabler
t,p	räknare

Efter inläsning av 10 värden $y(t), t = 1,2...$ 10. beräknade tidigare (III.2) påbörjas en loop svarande mot en iteration i a. Loopen genomlöpes 10 gånger. Härvid sker en beräkning analog med den tidigare beskrivna handräkningen. För varje loop skrivs det ut värden på förlustfunktionen V, första och andraderivatan av denna samt a.

INDATA

0,5774

0,4472

0,3780

0,3333

0,3015

0,2773

0,2582

0,2425

0,2294

0,2182

```
begin real a,x,dx,v,dv,ddv,h,r;
integer t,p;real array y[1:10];
real procedure g(X,A);real X,A;
g:=X/sqrt(1+2×A×X×X);
real procedure gx(X,A);real X,A;
gx:=1/sqrt(1+2×A×X×X)↑3;
real procedure ga(X,A);real X,A;
ga:=-(X/sqrt(1+2×A×X×X))↑3;
real procedure Newton(DV,DDV,A);real DV,DDV,A;
Newton:=A-DV/DDV;
for t:=1 step 1 until 10 do y[t]:=read;
a:=0.5;
write(↑EN,DIMENSION,V,DV,DDV,A↑);
for p:=1 step 1 until 10 do
  begin x:=1;dx:=0;v:=0;dv:=0;ddv:=0;
  for t:=1 step 1 until 10 do
    begin h:=gx(x,a);r:=ga(x,a);x:=g(x,a);
    dx:=h×dx+r;v:=v+(y[t]-x)↑2/2;
    dv:=dv-(y[t]-x)×dx;ddv:=ddv+dx×dx;
    end;
  a:=Newton(dv,ddv,a);punch(1);
  print(2,10,v);print(2,10,dv);
  print(2,10,ddv);print(3,10,a);
  end
end;
```

EN DIMENSION V DV DDVA

0.0574872941	-0.3553462200	1.1020962614	0.8224275708
0.0040769609	-0.0519707519	0.3313150263	0.9792896141
0.0000447709	-0.0043732581	0.2136094719	0.9997627609
0.0000000122	-0.0000555907	0.2026675874	1.0000370554
0.0000000046	-0.0000000086	0.2025261886	1.0000370973
0.0000000046	-0.0000000001	0.2025261668	1.0000370983
0.0000000046	-0.0000000001	0.2025261668	1.0000370992
0.0000000046	0.0000000003	0.2025261659	1.0000370973
0.0000000046	-0.0000000001	0.2025261668	1.0000370983
0.0000000046	-0.0000000001	0.2025261668	1.0000370992

V $\frac{dV}{da}$ $\frac{d^2V}{da^2}$ a

Resultat:

iteration	a
0	0,5
1	0,82243
2	0,97929
3	0,99976
4	1,0000371
5	"
.	"
.	"
.	"
10	"

Vi erhåller snabb konvergens med god noggrannhet. Exakta värden kan inte väntas bl.a. därför att vi inte har indata med större noggrannhet.

III.4

UPPSKATTNING AV NOGGRANNHETEN

Om vårt system vore lineärt, och om mätfelen y-x vore oberoende stokastiska variabler kunde noggrannheten i de gjorda koeff.-uppskattningarna skattas på samma sätt som i regressionsanalysen. Vi får då

$$\delta_i^2 = (\lambda^2 V_{aa}^{-1})_{ii} = (\lambda^2 \frac{d^2V}{da_i da_j}^{-1})_{ii}$$

$$\lambda^2 = \frac{2V(a)}{N}$$

Där N är antalet mätningar.

I detta fall är systemet olinеärt men det har visats av Box att ovanstående uttryck i alla fall ger en uppskattning av parameter-noggrannheten.

För vårt problem erhålles $\delta = 5 \cdot 10^{-5}$

IV.1

LÖSNING, ALLMÄNNA FALLET

Betrakta nu problemet mera allmänt. Låt x och a vara vektorer.

Analogt med III.1

$$\frac{dx}{dt}i = f_i(x_1(t)\ldots x_p(t),u(t),a_1\ldots a_v) \qquad i = 1,2,3\ldots.$$

$$y_i = g_i(x_1(t),\ldots x_p(t),u(t),a_1\ldots a_v)$$

Vi studerar i stället det diskreta fallet

$$x_i(t+1) = g_i(x_1(t)\ldots x_p(t),u(t),a_1\ldots a_v) \qquad t = 0,1,2\ldots..$$

Med vektorer

$$x(t+1) = g(x(t),u(t),a)$$

Inför förlustfunktionen

$$V(t+1) = \tfrac{1}{2}\sum_{s=1}^{t+1}\sum_{i=1}^{p}(y_i(t) - x_i(t))^2 = \tfrac{1}{2}\sum_{t=1}^{t+1}(y - x)^T(y - x)$$

För att beräkna förlustfunktionen och dess derivator förfares på ett sätt analogt med det i III.1. Vi anger först rekursiv formel för V(t).

$$V(t+1) = \tfrac{1}{2}(y(t+1) - x(t+1))^T(y(t+1) - x(t+1)) + V(t)$$

$$\frac{dV(t+1)}{da_j} = \frac{dV(t)}{da_j} - \frac{dx(t+1)}{da_j}^T(y(t+1) - x(t+1))$$

$$\frac{d^2V(t+1)}{da_v da_j} \approx \frac{d^2V(t)}{da_v da_j} + \frac{dx(t+1)}{da_j}^T \cdot \frac{dx(t+1)}{da_v}$$

Med $\frac{d^2V}{da_v da_j}$ menas den matris som innehåller andraderivatorna av förlustfunktionen V (följ. sid.). För detta behöver vi

$$\frac{dx(t+1)}{da_j} = \sum_{s=1}^{p}\frac{dg(t)}{dx_s} \cdot \frac{dx_s(t)}{da_j} + \frac{dg(t)}{da_j}$$

Även i detta allmännare fall är Newton-Raphsons lösningsmetod användbar (ref. 3).

Lösningen till ett system av typen

$$f_i(x_1, x_2 \ldots\ldots\ x_r) = 0 \qquad i = 1,2,3\ldots.$$

ges av

$$x_{n+1} = x_n - J \cdot f(x_n) \qquad n = 0,1,2\ldots.$$

om startvärdet x_0 valts så att konvergens erhålles.

Ovan är $x = (x_1, x_2, \ldots\ldots x_r)$ och J är av formen

$$J = \begin{pmatrix} \frac{df_1}{dx_1} & \frac{df_1}{dx_2} \ldots\ldots & \frac{df_1}{dx_r} \\ \frac{df_2}{dx_1} & \ldots. & \\ \vdots & & \\ \frac{df_r}{dx_1} & & \frac{df_r}{dx_r} \end{pmatrix}^{-1}$$

För $\frac{dV}{da_j} = 0$ fås

$$a_{k+1} = a_k - B \cdot \frac{dV}{da}$$

$$B = \begin{pmatrix} \frac{d^2V}{da_1^2} & \frac{d^2V}{da_2 da_1} & \ldots. & \frac{d^2V}{da_v da_1} \\ \vdots & & & \\ \frac{d^2V}{da_1 da_v} & \ldots\ldots & & \frac{d^2V}{da_v^2} \end{pmatrix}^{-1}_{x = a_k}$$

Här har förutsatts att x(0) är känd. Annars får även x(0) behandlas som okända parameterm $a = (a_1, a_2 \ldots\ldots a_v)$.

IV.2

PROGRAMMERING

Vi övergår till att formulera ett ALGOL-program som löser det allmänna problemet. Utformningen av programmet följer teorin som behandlats i IV.1 (jmf. även III).

Programmet är uppbyggt så att dess första och sista del är speciell för det exempel som just studeras. Den mellanliggande delen är däremot allmän för problem av typen i fråga. Denna allmänna del inleds av en serie procedurer nämligen Norm och INVPD samt NEWTON och FÖRLUST. Därefter kommer den återstående del som kunnats hållas helt generell. Programmets allmänna del betraktas först.

Program 2:

INVPD och Norm

Av föregående sida framgår att vi behöver en procedur som inverterar en matris. Matrisen ifråga är positivt definit vilket fått oss att välja en speciell rutin för inverteringen (ref.5). Rutinen förutsätter positivt defenit, symetrisk matris.

Procedurens formella parametrar är

n	matrisens ordning
a	matrisen
FAIL	läge
END	"

I proceduren finns en test av matrisen så att denna verkligen är positivt defenit. Om så inte är fallet skrives

detta ut och programmet fortsätter vid END.

För att inte ett för litet pivotelement skall användas jämföres detta med den minsta av maxnormerna för matrisen och matrisens transponat. Beräkning av normen sker i proceduren Norm. Om pivotelementet är för litet uppsöks läget FAIL.

NEWTON

Procedurens formella parametrar är

F	vektorn $f_i(x_1, x_2 \ldots x_r)$
FX	matrisen J
C	vektorn $(x_1, x_2 \ldots x_r)$
m	matrisens ordning

Proceduren utför beräkningar enligt N-R:s metod IV.1. D.v.s. där ingår endast en matrismultiplikation samt en subtraktion.

FÖRLUST

Formella parametrar

T	tidsvärde
DX	matris, derivatan av x efter a
F	matris, derivatan av g efter a
R	matris, derivatan av g efter x
V	förlustfunktionen
DV	vektor, förstaderivatan av V efter a
DDV	matris, andraderivatan av V efter a
Y	matris, mätvärden av vektorn y i olika tidpunkter
X	vektorn x
p1	komponenter i x
p2	komponenter i a

I denna procedur beräknas derivatan av x efter a samt förlustfunktionen V med första och andraderivata. Se kommentarer i programmet. Beräkningen sker med användande av de rekursiva formlerna i IV.1.

Den sista delen av det allmänna programmet utgörs av inläsning av begynnelsevärden från remsa i ordning

h

u(0)

a =(a_1, a_2 ...)

x =(x_1, x_2 ..) för t= 0

DX matris

Sedan inläses mätvärden enligt

u , y_1 , y_2, ... t = 1

" " " t = 2

o.s.v.

Efter läge nytt påbörjas loopen som sker för varje iteration, d.v.s. en gång föf varje a-värde.

För testerna av programmet hänvisas till appendix.

IV.3

EXEMPEL

Betrakta ekvationen

$$\dddot{x} + a_1\ddot{x} + a_2\dot{x} + a_3x^3 = b_1u + b_2u^2 \qquad a_1,a_2,a_3,b_1,b_2 \text{ okända}$$

Inför

$$\begin{cases} x_1 = x \\ x_2 = \dot{x} \end{cases}$$

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -a_1x_2 - a_2x_1 - a_3x_1^3 + b_1u + b_2u^2 \end{cases}$$

Detta approximeras med

$$\begin{cases} x_1(t+h) = (x_1 + hx_2)_t \\ x_2(t+h) = (x_2 + h(-a_1x_2 - a_2x_1 - a_3x_1^3 + b_1u + b_2u^2))_t \end{cases}$$

Då vi har mätvärden y_1,y_2 av x_1,x_2 vill vi bestämma vektorn

$a = (a_1,a_2,a_3,b_1,b_2)$
u insignal

h väljes här 0,1

Med tidigare beteckningar

$$\begin{cases} g_1 = hx_2 + x_1 \\ g_2 = (-a_1x_2 - a_2x_1 - a_3x_1^3 + b_1u + b_2u^2)h + x_2 \end{cases}$$

$$\begin{cases} \dfrac{dg_1}{dx_1} = 1 \qquad \dfrac{dg_1}{dx_2} = h \\ \dfrac{dg_2}{dx_1} = (-a_2 - 3a_3x_1^2)h \qquad \dfrac{dg_2}{dx_2} = -a_1h + 1 \end{cases}$$

$$\begin{cases} \dfrac{dg_1}{da_i} = 0 \qquad i = 1,2,3,4,5 \\ \dfrac{dg_2}{da_1} = -x_2h \;,\; \dfrac{dg_2}{da_2} = -x_1h \;,\; \dfrac{dg_2}{da_3} = -x_1^3h \;,\; \dfrac{dg_2}{db_1} = hu \;,\; \dfrac{dg_2}{db_2} = hu^2 \end{cases}$$

För att lösa problemet fordras kända mätvärden y(t). Hur det åstadkommes beskrives senare. Vi antar y(t) kända.
Första delen av ALGOL-programmet är avpassat för vårt problem. Det har varit nödvändigt eftersom dynamiska fält inte tillåts i SMIL-ALGOL (ref.2). De variabler som deklarerats i denna del är de samma som tidigare beskrivits, små bokstäver har använts utanför procedurerna.
I programmets sista del beräknas derivatorna av g efter x och a. Sedan anropas FÖRLUST varefter utskrift av V, DV och DDV sker. INVPD anropas, därefter NEWTON och ett nytt värde på vektorn a fås. Slutligen utskrives DDV-invers och vektorn a. Programmet går till läge NYTT om flera iterationer återstår.

```
begin commentI detta program uppskattas okända koef. i ett olinjärt system
med Newton-Raphsons metodik. Inläsning av h,u,a(begynnelsevärde),x,dx
radvis samt u och y i tidsföljd;
integer px,pa,q;
px:=2;pa:=5;q:=100;
   begin integer t1,s,l,w; real h,v,x1;
   real array u[0:100],y[1:2,1:100],a[1:5],x[1:2],dx[1:2,1:5],dv[1:5],
   ddv[1:5,1:5],f[1:2,1:2],r[1:2,1:5],x0[1:2],dx0[1:2,1:5];
   comment Här börjar allmänt program;;
```

```
procedure NEWTON(F,FX,C,m);
comment C itereras;value m;
integer m;real array F,FX,C;
  begin integer b,c;real d;
  for b:=1 step 1 until m do
    begin d:=0;
    for c:=1 step 1 until m do
    d:=d+FX[b,c]×F[c];
    C[b]:=C[b]-d;
    end
  end;

procedure FÖRLUST(T,DX,F,P,V,DV,DDV,Y,X,p1,p2);
comment Denna procedur beräknar derivatan av x efter a,förlustfunk.V
samt dennas första och andraderivata;
value p1,p2;integer p1,p2;real V, T;real array F,P,DV,DDV,DX,Y,X;
   begin integer i,k,j;real s;
   for i:=1step 1 until p1 do
   for k:=1 step 1 until p2 do
      begin s:=0;
      for j:=1 step 1 until p1 do
      s:=s+P[i,j]×DX[j,k]; DX[i,k]:=s+P[i,k]
      end DX;
   for i:=1step 1 until p1 do
   V:=V+(Y[i,T]-X[i])↑2/2;
   comment V klar;
   for i:=1 step 1 until p2 do
      begin s:=0;
      for j:=1 step 1 until p1 do
      s:=s+DX[j,i]×(Y[j,T]-X[j]); DV[i]:=DV[i]-s;
      end DV;
   for i:=1 step 1 until p2 do
   for k:=1 step 1 until p2 do
      begin s:=0;
      for j:=1 step 1 until p1 do
      s:=s+DX[j,i]×DX[j,k]; DDV[i,k]:=s+DDV[i,k];
      end DDV;
   end;;
```

```
    procedure Norm(A,n1,n2,s);
    comment Denna procedur beräknar en norm som är den minsta av maxnormerna för
            A och A:s transponat;
    integer n1,n2; real s; array A;
    begin integer i,j;real r,s1; s:=s1:=0;
        for j:=1step 1 until n2 do
        begin r:=0; for i:=1 step 1 until n1 do r:=r+abs(A[i,j]);
        if r>s1 then s1:=r
        end;
        for i:=1 step 1 until n1 do
        begin r:=0; for j:=1 step 1 until n2 do r:=r+abs(A[i,j]);
        if r>s then s:=r;
        end;
        if s>s1 then s:=s1
    end Slut på proceduren Norm;
procedure INVPD(t)ordning:(n)exit:(FAIL,END);
comment Med en variant av kvadratrotsmetoden inverterar proceduren en symmetrisk
    positivt definit matris av ordningen n. Om något pivotelement är mindre än 2-6
    lämnas proceduren och programmet fortsätter vid läge FAIL. Om t[1,1] är noll
    skrivs t ut och programmet fortsätter vid END;
value n; array t; integer n; label FAIL,END;
begin integer i,j,s; real array v[1:10]; real y,pivot,r;
for s:=0 step 1 until n-1 do

    begin if t[1,1]=0 then go to SINGULAR;
    pivot:=1/t[1,1]; Norm(t,n,n,r); if pivot<r×2-6 then go to FAIL;
        for i:=2 step 1 until n do v[i-1]:=t[1,i];
        for i:=1 step 1 until n-1 do
        begin t[i,n]:=y:=-v[i]×pivot;
        for j:=i step 1 until n-1 do t[i,j]:=t[i+1,j+1]+v[j]×y
        end;
        t[n,n]:=-pivot
    end;
for i:=1 step 1 until n do
for j:=i step 1 until n do t[i,j]:=t[j,i]:=-t[i,j];go to SLUT;
SINGULAR:punch(1);punch(1);
vrite(¤MATRISEN,ÄR,POSITIVT,DEFINIT¤);
for i:=1 step 1 until n do
    begin punch(1);for j:=1step 1 until n do print(-4,8,t[i,j])
    end;go to END;
SLUT:
end slut på proceduren INVPD;;
```

```
h:=read; u[0]:=read;
for s:=1 step 1 until pa do a[s]:=read;
for l:=1 step 1 until px do
    begin x0[l]:=read;
    for s:=1 step 1 until pa do
    dx0[l,s]:=read
    end;
for t1:=1 step 1 until q do
    begin u[t1]:=read;y[1,t1]:=read;y[2,t1]:=read
    end;
write('IDENTIFIERING,V,DV,DDV,DDVINV,A');
NYTT:for w:=1 step 1 until 0 do
    begin for l:=1 step 1 until px do
        begin x[l]:=x0[l];
        for s:=1 step 1 until pa do
        dx[l,s]:=dx0[l,s]
        end;
    v:=0;
    for s:=1 step 1 until pa do
        begin dv[s]:=0;
        for l:=1 step 1 until pa do
        ddv[s,l]:=0
        end HÄR slutar allmänt program;;
```

```
    for t1:=1 step 1 until q do
        begin f[1,1]:=1;f[1,2]:=h;
        f[2,1]:=(-a[2]-3×a[3]×x[1]↑2)×h;
    f[2,2]:=1-h×a[1];
        for l:=1 step 1 until pa do
        r[1,1]:=0;
        r[2,1]:=-h×x[2]; r[2,2]:=-h×x[1];
        r[2,3]:=-h×x[1]↑3;r[2,4]:=h×u[t1-1];
        r[2,5]:=h×u[t1-1]↑2;
        x1:=x[1]+h×x[2];
        x[2]:=x[2]+h×(-a[1]×x[2]-a[2]×x[1]-a[3]×x[1]↑3+
        a[4]×u[t1-1]+a[5]×u[t1-1]↑2); x[1]:=x1;
        PORLUSI(t1,dx,f,r,v,dv,ddv,J,x,px,pa)
        end;
punch(1);print(-3,10,v);punch(1);punch(1);
for s:=1 step 1 until pa do
print(-3,10,dv[s]);punch(1);
for s:=1 step 1 until pa do
    begin punch(1);
    for l:=1 step 1 until pa do
    print(-3,10,ddv[s,l])
    end;punch(1);
INVRD(ddv,pa,PEL,UT);
MNOZM(dv,ddv,a,pa);
for s:=1 step 1 until pa do
    begin punch(1);
    for l:=1 step 1 until pa do
    print(-3,10,ddv[s,l])
    end;
punch(1); punch(1);
for s:=1 step 1 until pa do
print(-3,10,a[s])
end iteration; go to UT;
PEL:write('PIVOTHELEMENTBYL<0-6>');
end;
UT:
end;
```

IV.4

GENERERING AV DATA

Till vårt program måste vi generera data. Vi betraktar åter exemplet i IV.3. Som insignal u väljer vi talen 1 och -1 slumpvis. Ett ALGOL-program har konstuerats som anropar en maskinkodad funktionsprocedur rand (ref. 4). Rand ger ett slumptal i intervallet (0,1). Om talet är större än $\frac{1}{2}$ sättes u = 1 annars sättes u = -1. Utskrift av 100 st. u fås på femkanalsremsa.

Därefter har ännu ett ALGOL-program gjorts som inläser de 100 u-värdena och härur beräknar x_1 och x_2 i motsvarande tidpunkt. I detta program inläses först antalet värde, i detta fall 100, så h-värdet, därefter a-vektorn samt begynnelsevärdena x_1, x_2, u för t = 0. Först sedan inläses de tidigare genererade u-värdena. Utskrift av u, x_1, x_2 erhålles på remsa i tidsföljd.

Indata:

100

0,1

0,2

0,2

0,2

0,4

0,3

0

0

1

```
begin comment Program för generering av 100 insignaler .De erhålles
med hjälp av en maskinkodad funktionsprocedur rand.Utskrift på remsa;
integer i; real x,u;
for i:=1 step 1 until 100 do
    begin x:=rand;
    if x>1/2 then u:=1 else u:=-1;
    stans(1); skrv(2,1,u)
    end;
stans(1);stans(8)
end;
```

-1.0	1.0
-1.0	1.0
-1.0	1.0
1.0	-1.0
-1.0	-1.0
-1.0	1.0
-1.0	-1.0
-1.0	-1.0
1.0	-1.0
-1.0	-1.0
1.0	1.0
1.0	-1.0
-1.0	1.0
1.0	-1.0
1.0	1.0
1.0	-1.0
1.0	-1.0
-1.0	-1.0
1.0	1.0
-1.0	-1.0
1.0	1.0
1.0	1.0
-1.0	-1.0
-1.0	1.0
1.0	-1.0
-1.0	1.0
-1.0	-1.0
-1.0	-1.0
-1.0	-1.0
-1.0	1.0
1.0	-1.0
-1.0	-1.0
-1.0	-1.0
1.0	1.0
1.0	1.0
1.0	-1.0
-1.0	-1.0
1.0	-1.0
1.0	-1.0
-1.0	1.0
1.0	1.0
1.0	1.0
1.0	-1.0
1.0	-1.0
-1.0	-1.0
1.0	1.0
1.0	-1.0
-1.0	1.0
	1.0
	1.0
	1.0
	1.0

```
begin comment Här genereras x då u och a kända,utskrift på remsa av u,x1 ,x2;
integer p,P;real h,u,x1;
real array a[1:5],x[1:2];
P:=read; h:=read;
for p:=1step 1 until 5 do a[p]:=read;
x[1]:=read; x[2]:=read; u:=read;
for p:=1 step 1 until P do
   begin
   x1:=x[1]+h×x[2];
   x[2]:=x[2]+h×(-a[1]×x[2]-a[2]×x[1]-a[3]×x[1]↑3+a[4]×u+a[5]×u↑2);
   x[1]:=x1; u:=read;
   stans(1); skrv(2,1,u);
   stans(1); skrv(-3,10,x[1]);
   stans(1); skrv(-3,10,x[2]);
   end;stans(1); stans(8)
end;
```

0

-1.0
0.0000000000 +00
7.0000000000 -02
-1.0
6.9999999962 -03
5.8600000031 -02
-1.0
1.2860000003 -02
4.728799317O -02
1.0
1.7589799325 -02
3.6084990780 -02
-1.0
2.1197298392 -02
1.0501140616 -01
-1.0
3.1698439009 -02
9.2487041577 -02
-1.0
4.0947143174 -02
8.0002694949 -02
-1.0
4.8947412669 -02
6.7582325153 -02
1.0
5.5705645196 -02
5.5249384976 -02
-1.0
6.1230583675 -02
1.2302682716 -01
1.0
7.3533266372 -02
1.0933708772 -01
1.0
8.4466975107 -02
1.7567172860 -01
-1.0
1.0203414801 -01
2.4045690167 -01
1.0
1.2607983816 -01
2.2358583528 -01
1.0
1.4843842163 -01
2.8655243813 -01
1.0

.

.

0.1.V.

IV.5

KÖRNING

Sedan erfordeliga indata genererats köres programmets ALGOL-del tillsammans med två dataremsor. Den första har data i den ordning som beskrivits i IV.2 och den andra innehåller u, x_1, x_2 (IV.4). Resultatet erhålles genom utskrift på radskrivare. Programmet är så utformat att antalet använda mätvärden kan varieras. SMIL:s kapacitet begränsar emellertid antalet u-värden till något mer än 400.

Indata:

0,1

1

0,3

0,25

0,15

0,45

0,2

0

0

0

0

0

0

0

0

0

0

0

Här följer andra remsan.

IDENTIFIERING V

V 5.6386225111

$\frac{dV}{da_i}$ 1.3525808975

$\frac{d^2V}{da_i da_j}$

6.3678736649
-3.8219114150
-3.1211929786
-2.7069195955
-9.7258626222

$\left(\frac{d^2V}{da_i da_j}\right)^{-1}$

1.1465011127
4.9694967642
-2.5593220200
1.3956636432
3.2199807111

a_1

V 2.7545119561
2.3333964683

$\frac{dV}{da_i}$ 4.0097471252

$\frac{d^2V}{da_i da_j}$

6.1494784653
-8.6674896329
-1.6769833154
-1.3986781351
-2.6679190844

$\left(\frac{d^2V}{da_i da_j}\right)^{-1}$

5.4658849500
-7.0457555428
3.7122331205
9.1068658382
1.9716133205

a_2

V 1.6797715361
1.1545442165

$\frac{dV}{da_i}$ -4.4936860911

$\frac{d^2V}{da_i da_j}$

2.3389383250
-6.4073040895
-1.4365072841
-6.2255900801
-1.4408536921

$\left(\frac{d^2V}{da_i da_j}\right)^{-1}$

3.6388226840
7.5104152933
3.8099769070
7.4113673300
1.5973452970

a_3

V 2.0764884500
6.6191673680

$\frac{dV}{da_i}$ 4.0932451590

$\frac{d^2V}{da_i da_j}$

1.6566921830
-7.9453289240
-6.2995653890
-4.9648608860
-6.3461643940

4.1528677120
3.1581530490
1.0920438360

8.1914;
1.7334;

a_4 1.9783;
6.9907;

-4.6580;

1.8090;
-4.2455;
-5.3778;
-5.7463;
-4.5549;

3.7779;
-1.4693;
1.3083;
7.7548;
1.4951;

a_5 2.0013;
4.5158;

3.7851;

1.7823
-4.7241
-5.4028
-5.6508
-4.6116

3.8141
-1.0063
1.2890
7.7979
1.5180

a_6 1.9998
4.2474

-3.6829

1.7843
-4.6869
-5.3996
-5.6584
-4.6031

3.8109
-1.0439
1.2904
7.7940
1.5160

a_7 2.0000
3.7716

3.4697

1.7841
-4.6906
-5.3998
-5.6577
-4.6039

3.811292
-1.040338
1.290342
7.794455
1.516245

a_8 1.999998

IV.6

RESULTAT

iter.	a_1	a_2	a_3	a_4	a_5
0	0,3	0,25	0,15	0,45	0,2
1	0,2755	0,3050	0,2483	0,3179	0,3260
2	0,1680	0,2478	0,1125	0,3948	0,2922
3	0,2076	0,2179	0,1782	0,3746	0,3009
4	0,1978	0,1994	0,2008	0,4004	0,2994
5	0,2001	0,2007	0,1999	0,3999	0,3000
6	0,199987	0,199995	0,200007	0,400013	0,299998
7	0,200001	0,200000	0,199999	0,399999	0,300000
8	0,200000	0,200000	0,200000	0,400000	0,300000

Resultatet är bra, och konvergensen snabb. Redan efter fem iterationer är tre siffror rätt och efter åtta iterationer är sex siffror korrekta.

Förlustfunktionen sjunker till det låga värdet 10^{-10}.

Derivatan av denna blir omkring 10^{-4}.

IV.7

UPPSKATTNING AV NOGRANNHETEN

Liksom i det enklare fallet (III.4) vill vi här uppskatta nogrannheten i resultatet. Approximativt erhålles i detta fall

$$\lambda^2 = \frac{2V(a)}{N}$$

$$\sigma_i^2 = (\lambda^2 V_{00}^{-1})_{ii}$$

Där N är antalet mätningar.

Idet behandlade fallet erhålles

$$\sigma_1 = 5 \cdot 10^{-7}$$
$$\sigma_2 = 9 \cdot 10^{-7}$$
$$\sigma_3 = 6 \cdot 10^{-7}$$
$$\sigma_4 = 12 \cdot 10^{-7}$$
$$\sigma_5 = 3 \cdot 10^{-7}$$

KONVERGENS

Beträffande konvergens vid Newton-Raphsons metod hänvisas till ref.3.

Ett försök med nytt startvärde a_0 som endast obetydligt avvek från det först använda (IV.5) gav nedanstående resultat.

iter.	a_1	a_2	a_3	a_4	a_5
0	0,3	0,3	0,2	0,5	0,15
1	0,29	0,38	0,31	0,30	0,34
2	0,23	0,34	-0,09	0,40	0,31

Här avbröts räkningen på grund av spill. Vi ser att a_4 och a_5 konvergerar snabbt, a_1 också tämligen medan a_3 som utgick från rätt värde helt tycks förstöra resultatet.

Detta uppförande kan eventuellt bero på att med våra a-värde kommer systemet att ligga myck et nära stabilitetsområdets rand. Stabilitetsområdet påverkas främst av parametern a_1.

IDENTIFIERING

V 1.007966070

$\frac{dV}{da_i}$ 9.685009352

$\frac{d^2V}{da_i da_j}$

4.026656534
-2.748955231
-1.042647362
-2.339922096
-6.787349179

$\left(\frac{d^2V}{da_i da_j}\right)^{-1}$

1.424021788
5.282722547
-5.695890087
1.479524111
2.535211069

a_1 2.884172696
5.115895211

V

$\frac{dV}{da_i}$ -1.686786308

$\frac{d^2V}{da_i da_j}$

3.934152670
-5.853111539
-8.722935363
-2.604906268
-2.439598480

$\left(\frac{d^2V}{da_i da_j}\right)^{-1}$

4.718055579
2.295268902
3.243578700
6.183953609
1.634281894

a_2 2.285791786

APPENDIX

TEST AV PROGRAM OCH PROGRAMDELAR

Innan program 2 sammansattes av sina olika delar testades dessa. För testen av proceduren NEWTON användes

$$e^{-x} - \sin x = 0$$

$$x_0 = 0,6$$

Exemplet hämtades från ref.1. Där angavs följande värden på x vid iterationen

$$x_1 = 0,5885$$

$$x_2 = 0,58853274$$

Vid körning av NEWTON i ett enkelt testprogram erhölls en utskrift helt överensstämmande med värdena ovan. Då en test i fallet x vektor inte genomförts har i stället valts att för hand kontrollera resultatet av första iterationen vid körningen av program 2. Handräkningen gav god överensstämmelse med det tidigare resultatet.

```
begin comment test av Newton;
integer k; real array f,x[1:1],fx[1:1,1:1];
procedure NEWTON(F,FX,C,m);
comment C itereras;value m;
integer m;real array F,FX,C;
   begin integer b,c;real d;
   for b:=1 step 1 until m do
      begin d:=0;
      for c:=1 step 1 until m do
      d:=d+FX[b,c]*F[c];
      C[b]:=C[b]-d;
      end
   end;
x[1]:=0.6;
for k:=1 step 1 until 4 do
   begin f[1]:=exp(-x[1])-sin(x[1]);
   fx[1,1]:=-1/(exp(-x[1])+cos(x[1]));
   NEWTON(f,fx,x,1);
   write(¤NEST,AV,NEWTON¤);
   print(2,10,x[1]); punch(1);
   end
end;
```

```
TEST AV NEWTON   0.5884795193
TEST AV NEWTON   0.5885327435
TEST AV NEWTON   0.5885327444
TEST AV NEWTON   0.5885327444
```

För test av proceduren FÖRLUST användes samma ekvationer som vid körningen av program 2, d.v.s. i diskret form

$$x_1(t+h) = x_1(t) + hx_2(t)$$

$$x_2(t+h) = x_2(t) + h(-a_1x_2(t) - a_2x_1(t) - a_3x_1^3(t) + b_1u(t) + b_2u(t)^2)$$

Här väljes $h=a_1=a_2=a_3=a_4=a_5=1$

$x_1(0)=x_2(0)=1$

$u(0) = 1$

$u(1) = 0$

$u(2) = -1$

$u(3) = 1$

$u(4) = -1$

$u(5) = 0$

Det ger mätvärdena

t	y_1	y_2
0	0	0
1	2	0
2	2	-10
3	-8	-10
4	-18	522
5	504	5850

Gissa ett a-värde $a_1=a_2=a_3=a_4=a_5=\frac{1}{2}$

Med formlerna från IV.1 och IV.3 beräknas V = 0,125

$$DV = \begin{pmatrix} -0,5 \\ -0,5 \\ -0,5 \\ 0,5 \\ 0,5 \end{pmatrix} \qquad DDV = \begin{pmatrix} 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 \end{pmatrix}$$

FÖRLUST köres i testprogrammet med data i ordning:

$x_1(1), x_2(1)$

$y_1(1), y_2(1)$

$y_1(2), \ldots\ldots$

...

$\frac{dg_1}{dx_k}$ k=1,2

$\frac{dg_2}{dx_k}$ k=1,2

$\frac{dg_1}{da_k}$ k=1,2,3,4,5

$\frac{dg_2}{da_k}$ k=1,2,3,4,5

En utskrift erhålles överensstämmande med de beräknade värdena.

FÖRLUST fungerar tydligen bra.

```
begin comment Test av FÖRLUST,f,r stansas radvis på remsa,y stansas tidsföljd;
integer m,n;real v;real array dx[1:2,1:5],x[1:2],dv[1:5],ddv[1:5,1:5],
f[1:2,1:2],r[1:2,1:5],y[1:2,1:3];
procedure FÖRLUST(T,DX,F,R,V,DV,DDV,Y,X,p1,p2);
comment Denna procedur beräknar derivatan av x efter a,förlustfunk.V
samt dennas första och andraderivata;
value p1,p2;integer p1,p2;real V, T;real array F,R,DV,DDV,DX,Y,X;
   begin integer i,k,j;real s;
   for i:=1step 1 until p1 do
   for k:=1 step 1 until p2 do
      begin s:=0;
      for j:=1 step 1 until p1 do
      s:=s+F[i,j]×DX[j,k]; DX[i,k]:=s+R[i,k]
      end DX;
   for i:=1step 1 until p1 do
   V:=V+(Y[i,T]-x[i])↑2/2;
   comment V klar;
   for i:=1 step 1 until p2 do
      begin s:=0;
      for j:=1 step 1 until p1 do
      s:=s+DX[j,i]×(Y[j,T]-X[j]); DV[i]:=DV[i]-s;
      end DV;
   for i:=1 step 1 until p2 do
   for k:=1 step 1 until p2 do
      begin s:=0;
      for j:=1 step 1 until p1 do
      s:=s+DX[j,i]×DX[j,k]; DDV[i,k]:=s+DDV[i,k];
      end DDV;
   end;;
```

```
v:=0;
for m:=1 step 1 until 2 do
    begin x[m]:=read;
    for n:=1 step 1 until 5 do
    dv[m,n]:=0
    end;
for m:=1 step 1 until 5 do
    begin dv[m]:=0;
    for n:=1 step 1 until 5 do
    ddv[m,n]:=0
    end;
for m:=1step 1 until 3 do
for n:=1 step 1 until 2 do
y[n,m]:=read;
for m:=1 step 1 until 2 do
for n:=1 step 1 until 2 do
f[m,n]:=read;
for m:=1 step 1 until 2 do
for n:=1 step 1 until 5 do
r[m,n]:=read;
vrite('TEST,AV,FÖRLUST,V,DV,DDV');
FÖRLUST(1,dx,f,r,v,dv,ddv,y,x,2,5);
punch(1);print(2,8,v);
punch(1);
for m:=1step 1 until 5 do
print(2,3,dv[m]);punch(1);
for m:=1 step 1 until 5 do
    begin punch(1);
    for n:=1 step 1 until 5 do
    print(2,3,ddv[m,n])
    end
end;
```

Data till test av MÖRLUND

```
2
0.5
2
0
2
-10
-.8
-.10
1
1
2
0.5
0
0
0
0
0
-1
-1
-1
1
1
*
```

TEST AV FÖRLUST V DV DDV

V 0.12500000

DV

	-0.500	-0.500	-0.500	0.500	0.500
	1.000	1.000	1.000	-1.000	-1.000
DDV	1.000	1.000	1.000	-1.000	-1.000
	1.000	1.000	1.000	-1.000	-1.000
	-1.000	-1.000	-1.000	1.000	1.000
	-1.000	-1.000	-1.000	1.000	1.000

En kontroll av genereringsprogrammet (IV.4) har gjorts med användande av samma exempel som i föregående test.

Indata: (se IV.4)

5
1
1
1
1
1
1
1
1
1
1
0
-1
1
-1
0

Utskrift av u, x_1, x_2 enligt följande sida erhölls. Värdena överensstämmer med de som beräknats i samband med test av FÖRLUST.

Test av generering, utskrift.

```
 0.0
       2.0000000000
       0.0000000000
-1.0
       2.0000000000
     -10.0000000000
 1.0
      -8.0000000000
     -10.0000000000
-1.0
     -18.0000000000
     522.0000000000
 0.0
     504.0000000000
    5850.0000000000
```

En slutlig kontroll av program 2 i sin helhet har gjorts genom att läsa in a-värdena

0,2
0,2
0,2
0,4
0,3

Då utskrift av de i programmet bildade x_1 och x_2 samt V önskades omformades del fem av programmet enligt nedan. En utskrift överensstämmande med den i IV.4 erhölls. Förlustfunktionen var approximativt noll.

```
    for t1:=1 step 1 until q do
        begin f[1,1]:=1;f[1,2]:=h;
        f[2,1]:=(-a[0]-3×a[3]×x[1]↑2)×h;
    f[2,2]:=1-h×a[1];
        for l:=1 step 1 until pa do
        r[1,1]:=0;
        r[2,1]:=-h×x[0]; r[2,2]:=-h×x[1];
        r[2,3]:=-h×x[1]↑3;r[2,4]:=-h×x[t1-1];
        r[2,5]:=-h×x[t1-1]↑2;
        x1:=x[1]+h×x[2];
        x[2]:=x[2]+h×(-a[1]×x[2]-a[0]×x[1]-a[3]×x[1]↑3-
        a[4]×x[t1-1]-a[5]×x[t1-1]↑2); x[1]:=x1;
        punch(1),print(-3,10,x[1]);print(-3,10,x[0]);
        PCHUSM(t1,dx,f,r,v,dv,ddv,y,x,pa,pa);print(-3,10,v)
        end;go to UT;
punch(1);print(-3,10,v);punch(1);punch(1);
for s:=1 step 1 until pa do
print(-3,10,dv[s]),punch(1);
for s:=1 step 1 until pa do
    begin punch(1);
    for l:=1 step 1 until pa do
    print(-3,10,ddv[s,l])
    end;punch(1);
INVPD(ddv,pa,VAL,UT);
MULTOU(dv,ddv,s,pa);
for s:=1 step 1 until pa do
    begin punch(1);
    for l:=1 step 1 until pa do
    print(-3,10,ddv[s,l])
    end;
punch(1); punch(1);
for s:=1 step 1 until pa do
print(-3,10,a[s])
end iteration; go to UT;
VAL:vrite(PIVOTELEMENT=0);
end;
UT:
end;;
```

[illegible] TEST

1.1

1.3
1.2
1.5
1.4
1.3
1
1
1
1
1
1
1
1
1
1
*

X_1 X_2 V

IDENTIFIERING V DV DDV DDVINV A

$0.0000000000_{10}+0$	$7.0000000000_{10}-2$	$2.7105054315_{10}-20$
$6.9999999962_{10}-3$	$5.8600000031_{10}-2$	$4.2675902172_{10}-20$
$1.2860000003_{10}-2$	$4.7287993170_{10}-2$	$5.0405077822_{10}-20$
$1.7588799325_{10}-2$	$3.6084990780_{10}-2$	$5.8875407278_{10}-20$
$2.1197298392_{10}-2$	$1.0501140616_{10}-1$	$8.7674527466_{10}-20$
$3.1698439009_{10}-2$	$9.2487041577_{10}-2$	$1.1859123017_{10}-19$
$4.0947143174_{10}-2$	$8.0002694949_{10}-2$	$1.5247254800_{10}-19$
$4.8947412669_{10}-2$	$6.7582325153_{10}-2$	$1.7830705298_{10}-19$
$5.5705645196_{10}-2$	$5.5249384976_{10}-2$	$2.0880023911_{10}-19$
$6.1230583675_{10}-2$	$1.2302682716_{10}-1$	$2.6639847960_{10}-19$
$7.3533266372_{10}-2$	$1.0933708772_{10}-1$	$3.2060858812_{10}-19$
$8.4466975107_{10}-2$	$1.7567172860_{10}-1$	$4.2394660785_{10}-19$
$1.0203414801_{10}-1$	$2.4045690167_{10}-1$	$7.1024374403_{10}-19$
$1.2607983816_{10}-1$	$2.2358583528_{10}-1$	$8.7965033352_{10}-19$
$1.4843842163_{10}-1$	$2.8655243813_{10}-1$	$1.2320160400_{10}-18$
$1.7709366548_{10}-1$	$3.4778720717_{10}-1$	$2.0180626157_{10}-18$
$2.1187238618_{10}-1$	$4.0717850923_{10}-1$	$2.5601637009_{10}-18$
$2.5259023718_{10}-1$	$4.6460727266_{10}-1$	$3.4817355480_{10}-18$
$2.9905096441_{10}-1$	$4.3994100801_{10}-1$	$4.1593619063_{10}-18$
$3.4304506517_{10}-1$	$4.9462627731_{10}-1$	$5.0267236456_{10}-18$
$3.9250769279_{10}-1$	$4.6706546023_{10}-1$	$6.1380308717_{10}-18$
$4.3921423889_{10}-1$	$5.1866458468_{10}-1$	$7.5474936962_{10}-18$
$4.9108069725_{10}-1$	$5.6781243905_{10}-1$	$9.4990576058_{10}-18$
$5.4786194115_{10}-1$	$5.3426599353_{10}-1$	$1.1450621522_{10}-17$
$6.0128854066_{10}-1$	$4.9933459013_{10}-1$	$1.3402185430_{10}-17$
$6.5122199989_{10}-1$	$5.4297423511_{10}-1$	$1.5353749338_{10}-17$
$7.0551942326_{10}-1$	$5.0356677472_{10}-1$	$1.7522153686_{10}-17$
$7.5587610080_{10}-1$	$4.6236149705_{10}-1$	$1.9175562001_{10}-17$
$8.0211225003_{10}-1$	$4.1935936883_{10}-1$	$2.1343966349_{10}-17$
$8.4404818713_{10}-1$	$3.7460861168_{10}-1$	$2.4488152656_{10}-17$
$8.8150904849_{10}-1$	$3.2820918466_{10}-1$	$2.7442603576_{10}-17$
$9.1432996690_{10}-1$	$3.6031514313_{10}-1$	$3.1779412254_{10}-17$
$9.5036148130_{10}-1$	$3.0953465700_{10}-1$	$3.4923598561_{10}-17$
$9.8131494671_{10}-1$	$2.5716965273_{10}-1$	$3.9070671871_{10}-17$
$1.0070319119_{10}+0$	$2.8350024651_{10}-1$	$4.3217745199_{10}-17$
$1.0353819364_{10}+0$	$3.0726471468_{10}-1$	$4.5386149547_{10}-17$
$1.0661084083_{10}+0$	$3.2821286674_{10}-1$	$4.9722958207_{10}-17$
$1.0989296948_{10}+0$	$2.6609193924_{10}-1$	$5.1701627187_{10}-17$
$1.1255388883_{10}+0$	$2.8224913515_{10}-1$	$5.5848700515_{10}-17$
$1.1537638017_{10}+0$	$2.9557587094_{10}-1$	$5.9995773807_{10}-17$
$1.1833213884_{10}+0$	$2.2587194126_{10}-1$	$6.4068308211_{10}-17$
$1.2059085825_{10}+0$	$2.3454917091_{10}-1$	$6.8215381540_{10}-17$
$1.2293634992_{10}+0$	$2.4066699650_{10}-1$	$7.2362454831_{10}-17$
$1.2534301988_{10}+0$	$2.4410679452_{10}-1$	$7.6434989273_{10}-17$
$1.2778408778_{10}+0$	$2.4477109014_{10}-1$	$8.3705920055_{10}-17$
$1.3023179871_{10}+0$	$1.6258770357_{10}-1$	$8.7670034244_{10}-17$
$1.3185767577_{10}+0$	$1.5911412648_{10}-1$	$9.4669914543_{10}-17$
$1.3344881702_{10}+0$	$1.5370957981_{10}-1$	$9.8634028732_{10}-17$
$1.3498591277_{10}+0$	$6.6414928175_{10}-2$	$1.0256426166_{10}-16$
$1.3565006209_{10}+0$	$5.8897349834_{10}-2$	$1.0952390786_{10}-16$
$1.3623903561_{10}+0$	$5.0667619146_{10}-2$	$1.1647804835_{10}-16$
$1.3674571178_{10}+0$	$4.1831600964_{10}-2$	$1.2342753019_{10}-16$
$1.3716402780_{10}+0$	$-4.7495400495_{10}-2$	$1.2734590461_{10}-16$
$1.3668907377_{10}+0$	$-1.3559029754_{10}-1$	$1.3434578487_{10}-16$
$1.3533317083_{10}+0$	$-1.4129401398_{10}-1$	$1.4134566513_{10}-16$
$1.3392023071_{10}+0$	$-2.2510748989_{10}-1$	$1.4541819952_{10}-16$
$1.3166915578_{10}+0$	$-3.0542557705_{10}-1$	$1.5260103894_{10}-16$
$1.2861489998_{10}+0$	$-3.8130524493_{10}-1$	$1.5718179307_{10}-16$
$1.2480184752_{10}+0$	$-4.5195253975_{10}-1$	$1.6176254730_{10}-16$
$1.2028232216_{10}+0$	$-4.3675088472_{10}-1$	$1.6634330144_{10}-16$
$1.1591481333_{10}+0$	$-4.9687683209_{10}-1$	$1.7122221123_{10}-16$
$1.1094604497_{10}+0$	$-4.7127145230_{10}-1$	$1.7610112102_{10}-16$
$1.0623333044_{10}+0$	$-5.2134798467_{10}-1$	$1.7881162650_{10}-16$
$1.0101985055_{10}+0$	$-4.8614565953_{10}-1$	$1.8369053620_{10}-16$

x_1	x_2	v
$9.6158393993_{10}-1$	$-5.2724488861_{10}-1$	$1.8737682364_{10}-16$
$9.0885945111_{10}-1$	$-5.6371411979_{10}-1$	$1.9106311099_{10}-16$
$8.5248803943_{10}-1$	$-5.9563184753_{10}-1$	$1.9377361647_{10}-16$
$7.9292485453_{10}-1$	$-5.4315964430_{10}-1$	$1.9745990382_{10}-16$
$7.3860889039_{10}-1$	$-5.6812565810_{10}-1$	$2.0017040930_{10}-16$
$6.8179632462_{10}-1$	$-5.0959418267_{10}-1$	$2.0233881361_{10}-16$
$6.3083690628_{10}-1$	$-4.4937683418_{10}-1$	$2.0475116353_{10}-16$
$5.8589922301_{10}-1$	$-4.6802693232_{10}-1$	$2.0670272745_{10}-16$
$5.3909653015_{10}-1$	$-4.0440690331_{10}-1$	$2.0811219010_{10}-16$
$4.9865584000_{10}-1$	$-4.2023419514_{10}-1$	$2.0946744289_{10}-16$
$4.5663242042_{10}-1$	$-3.5428251978_{10}-1$	$2.1057875007_{10}-16$
$4.2120416834_{10}-1$	$-3.6823379527_{10}-1$	$2.1193400286_{10}-16$
$3.8438078872_{10}-1$	$-3.8078774418_{10}-1$	$2.1280136462_{10}-16$
$3.4630201440_{10}-1$	$-3.9199543949_{10}-1$	$2.1391267180_{10}-16$
$3.0710247047_{10}-1$	$-3.2191217690_{10}-1$	$2.1459029819_{10}-16$
$2.7491125278_{10}-1$	$-3.3219525124_{10}-1$	$2.1526792459_{10}-16$
$2.4169172775_{10}-1$	$-3.4146510623_{10}-1$	$2.1587101202_{10}-16$
$2.0754521694_{10}-1$	$-3.4975200667_{10}-1$	$2.1628436408_{10}-16$
$1.7257001632_{10}-1$	$-2.7708667106_{10}-1$	$2.1663672979_{10}-16$
$1.4486134927_{10}-1$	$-2.0509912203_{10}-1$	$2.1680613644_{10}-16$
$1.2435143711_{10}-1$	$-2.1395516432_{10}-1$	$2.1701789461_{10}-16$
$1.0295592071_{10}-1$	$-2.2220154739_{10}-1$	$2.1721440628_{10}-16$
$8.0735765993_{10}-2$	$-2.2983846124_{10}-1$	$2.1741091795_{10}-16$
$5.7751919887_{10}-2$	$-2.3686693254_{10}-1$	$2.1759557109_{10}-16$
$3.4065226651_{10}-2$	$-1.6328848460_{10}-1$	$2.1770780291_{10}-16$
$1.7736378200_{10}-2$	$-9.0704810023_{10}-2$	$2.1775110755_{10}-16$
$8.6658971905_{10}-3$	$-1.9245553007_{10}+2$	$2.1775322500_{10}-16$
$6.7413418926_{10}-3$	$-2.9033972881_{10}-2$	$2.1775727495_{10}-16$
$3.8379446025_{10}-3$	$-3.8588126357_{10}-2$	$2.1776415705_{10}-16$
$-2.0868032146_{10}-5$	$-4.7893123850_{10}-2$	$2.1777474507_{10}-16$
$-4.8101804181_{10}-3$	$2.3065155986_{10}-2$	$2.1777749788_{10}-16$
$-2.5036648176_{10}-3$	$1.2700058715_{10}-2$	$2.1777849216_{10}-16$
$-1.2336589470_{10}-3$	$8.2496131137_{10}-2$	$2.1780560389_{10}-16$
$7.0159541666_{10}-3$	$1.5087088178_{10}-1$	$2.1786682847_{10}-16$
$2.2103042341_{10}-2$	$2.1771313827_{10}-1$	$2.1803888194_{10}-16$
$4.3874356150_{10}-2$	$2.8291659876_{10}-1$	$2.1829341538_{10}-16$

REFERENSER

1. Fröberg, C-E Lärobok i numerisk analys

2. Ekman, T Större PM angående SMIL-ALGOL

3. Saaty, Nonlinear Mathematics

4. Eriksson, K-E Numerisk bestämning av processdynamik

5. Collected Algorithms from CACM