Optimering medelst linear sökning - ett tidsdiskret Simnon system

Axelsson, Jan Peter

1987

Document Version:
Förlagets slutgiltiga version

CODEN: LUTFD2/(TFRT-7361)/1-9/(1987)

Optimering medelst linear sökning - ett tidsdiskret simnon system

Jan Peter Axelsson

Department of Automatic Control
Lund Institute of Technology
May 1987

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
INTERNAL REPORT

Date of issue
May 1987

Document Number
CODEN: LUTFD2/(TFRT-7361)/1-9/(1987)

Author(s)
Jan Peter Axelsson

Supervisor

Sponsoring organisation

Title and subtitle
Optimering medelst linear sökning - ett tidsdiskret SIMNON system. (Optimization using linear search - a time discrete SIMNON system).

Abstract

In this report is presented a simnon system SEARCH which optimizes one parameter. The method used is Fibonacci and Golden section search. It means that only the value of the loss function is used and not any properites of its partiell derivatives. Therefore it need not be differentiable. However, a fundamental requirement is that the loss function has only one minima.

The system SEARCH is written mainly for the PC-version of simnon. In the VAX-version of simnon there is already a standard system OPTA written in FORTRAN which solves the task.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
9

Recipient's notes

Security classification

OPTIMERING AV REGULATORPARAMETER MEDELST LINEAR SÖKNING

Här presenteras ett simnon system SEARCH som kan trimma en regulator parameter. Trimningen görs utifrån egenskaper i tidsplandet hos det slutna systemet. Ett stegsvar görs och en förlustfunktion evalueras. Metoden förutsätter att förlustfunktionen har endast ett lokalt minima. Inga andra egenskaper hos förlustfunktionen utnyttjas i algoritmen.

Systemet SEARCH är i första hand skriven med tanke på PC-versionen av simnon. Då denna version inte tillåter externa FORTRAN system, är SEARCH skriven som ett tidsdiskret simnon system. I VAX-versionen av simnon ingår redan ett standard system OPTA skrivet i FORTRAN, som löser det här problemet (också för flera variabler). SEARCH kan förstås, precis som OPTA, användas till andra exempel där optimering skall göras.

Ett exempel

För att illustrera algoritmen ges här ett enkelt exempel. En motor återkopplas proportionellt och stegsvaret utvärderas enligt ITAE kriteriet. I Fig.1. visas resultatet av en optimering.

Process: $y(t) = \frac{1}{p(p+1)}\, u(t)$

Regulator: $u(t) = k \cdot (y_r(t) - y(t))$

Förlustfunktion: $J(t) = \int_0^t \tau \cdot (y_r(\tau) - y(\tau))\, d\tau$

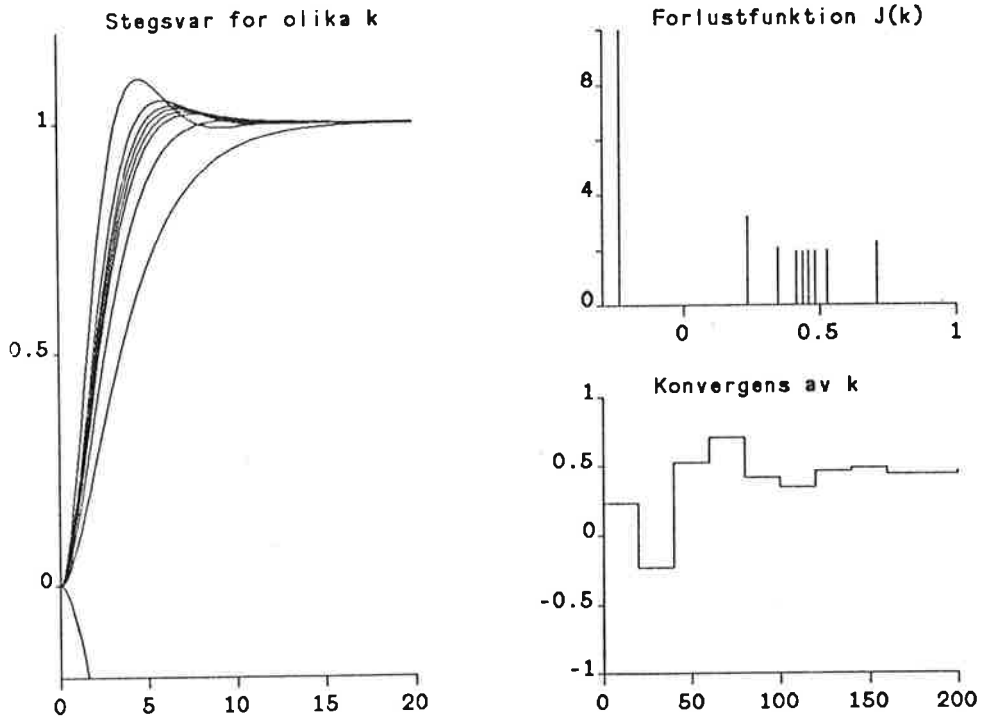

Fig. 1. Till vänster ses stegsvaret för olika värden på k under trimningen. Överst till höger är de utvärderingar av förlustfunktionen som gjorts och nederst till höger visas insvängningen mot optimalt k värde. Lägg märke till, att för negativt k värde var systemet instabilt.

Idén bakom algoritmen

Algoritmen utnyttjar att funktionen som skall minimeras J(P) endast har ett lokalt minima i det intervall där minima skall sökas (J unimodal). Vidare användns ej någon kunskap om derivator av J(·). Endast evalueringar av J(·) i några punkter förutsättes vara möjligt och målet är att behöva göra så få funktionsevalueringar som möjligt för att få Pmin med viss önskad precision.

Utgå från tre punkter i ett J(P) diagram: ändpunkter i ett intervall samt en punkt i intervallet. Bilda en fjärde punkt. Utifrån denna fjärde punkt kan nu det intervall där minima måste vara, minskas. Betrakta Fig. 2.

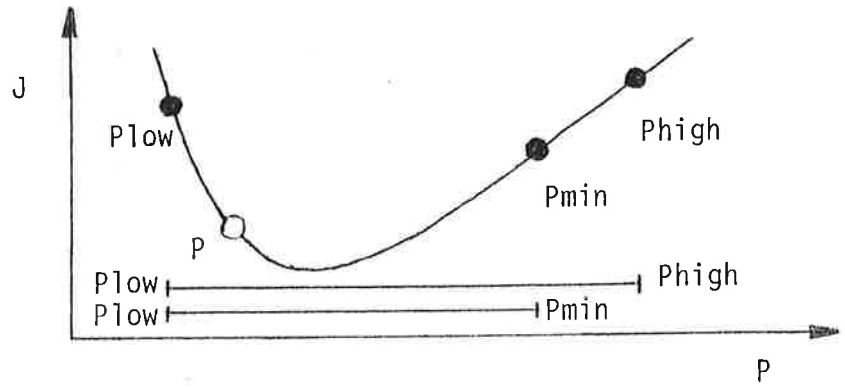

Fig. 2. Figuren visar den princip algoritmen bygger på. Ursprungligen är osäkerhetsintervallet [Plow, Phigh] och Pmin är punkten för minsta värdet hitills. Då det nya värdet J(P) inkommer kan osäkerhetsintervallet minskas. Ifall J(P)<Jmin måste minimat ligga i intervallet [Plow, Pmin] annars i [P, Phigh].

Algoritmen har fem tillstånd. Två tillstånd för att representera aktuellt osäkerhetsintervall [Plow, Phigh]. Minsta värde hitills i intervallet, Jmin, och motsvarande punkt Pmin, utgör ytterliggare två tillstånd. Ett femte tillstånd, P, krävs för att lagra undan utsignalen från algoritmen, Ptest, till dess att den är evaluerad och motsvarande J värde är tillgänglig på insignalen.

Uppdatering av tillstånden

Uppdateringen av Plow, Phigh och Pmin styrs av två villkor: huruvida J(P)<Jmin, och om P är till vänster eller höger om mitten, Pm, av det intervall där Jmin sökes. Följande uppdateringsregler erhålles:

J(P) < Jmin		J(P) > Jmin	
P < Pm	P > Pm	P < Pm	P > Pm
Plow = Plow	Plow = Pmin	Plow = Plow	Plow = P
Phigh = Pmin	Phigh = Phigh	Phigh = P	Phigh = Phigh
Pmin = P	Pmin = P	Pmin = Pmin	Pmin = Pmin

Uppdateringen av Pmin beror bara av J(P) och Jmin och är enkel att skriva i simnon. Däremot är uppdateringen av Plow och Phigh något mer komplicerad. Utnyttjas operationen exklusivt-eller, mellan första och andra logiska villkoret ovan, förenklas uppdateringsreglerna betydligt.

Uppstart och val av initialtillstånd

Vid uppstart skall Plow och Phigh ges värden samt den nya test punkten P som är utsignal från SEARCH. För att få ett initialvärde till Jmin måste en simulering göras och detta görs med initialvärdet på P. Det är då naturligt att också sätta Pmin till P initialt. Se vidare under rubriken: kodning i simnon.

Var minimat ligger i intervallet [Plow, Phigh] påverkar inte konvergens hastigheten. Däremot är valet av startpunkt P kritiskt för hur snabbt konvergens fås. För vissa initial värden på P når algoritmen stationaritet innan konvergens nåtts. Detta är lätt att förstå. Fenomenet inträffar då test punkten P hamnar mitt i aktuellt intervall [Plow, Phigh]. Nästa testpunkt skall enligt regeln läggas symmetriskt med punkten redan i detta intervall, och hamnar då i samma punkt som tidigare.

En enkel analys visar att osäkerhetsintervallet [Plow, Phigh] följer en differensekvation besläktad med den rekursiva ekvationen för Fibonacci talen. Faktum är att skall vi nå snabbast möjliga konvergens på N steg skall initialvärdet på P sättas till F_{N-1}/F_N där F_N är just Fibonacci talen. Typiskt är att algoritmen med ett sådant initialvärde stannar efter N steg och om N är litet kan osäkerhetsintervallet fortfarande vara stort. Önskas ett litet osäkerhetsintervall måste N väljas stort. Låter vi $N \rightarrow \infty$, ger det $F_{N-1}/F_N \rightarrow 2/(1+\sqrt{5}) \approx 0.618034$ (gyllene snittet). Detta initialvärde garanterar konvergens mot ett litet osäkerhetsintervall och konvergensen går nästan lika snabbt som då initialvärdet väljes för N = 3, 4 ... . Ett speciellt fenomen inträffar då initialvärdet väljes till ett tal, nära ett tal för små N. Då konvergerar osäkerhetsintervallet först snabbt de första N stegen och sedan mycket, mycket långsamt.

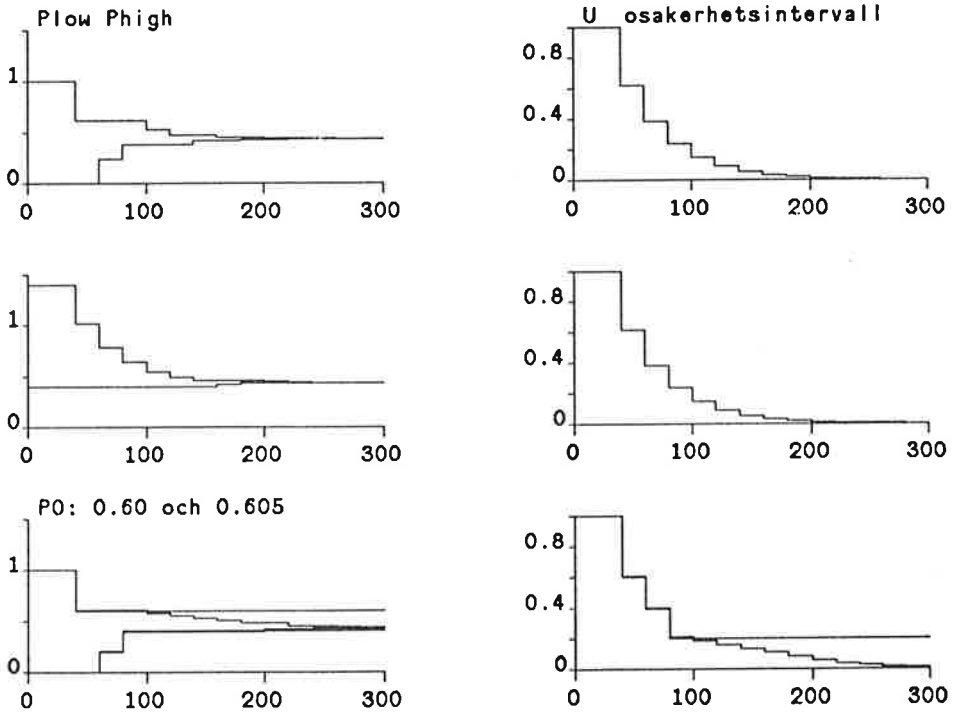

Fig. 3. Här visas fyra optimeringar av regulatorparametern, till motorn i ovan angivna exempel. Till vänster visas osäkerhetsintervallet [Plow, Phigh] och till höger storleken på intervallet U=Phigh-Plow. Överst visas konvergens då minimat ligger i mitten. I mellersta figuren är Plow och Phigh initialt förskjutna så att minimat ligger nära Plow. Startpunkten är vald för konvergens för stora N i dessa båda optimeringar. I nedersta figuren är startpunkten vald för N=3. I samma figur visas effekten av en liten störning av läget på startpunkten.

Något om kodning i simnon

Process och regulator har här förts samman till ett tidskontinuerligt system PROCESS medan optimeringsalgoritmen har lagts i ett tidsdiskret system SEARCH. Connecting system är TUNE. Förutom optimering av regulatorparameter, styrs: start, stop och nollställning, av process och regulator från SEARCH. Det är naturligt att ha två tidsbegrepp: ett för det totala simuleringstiden [t] och ett för tiden i ett stegsvar [tau]. Detta för att förlustfunktionen skall kunna få bero av tiden under ett stegsvar. Vidare kan det vara en fördel att kunna plotta olika stegsvar i samma diagram med olika regulatorparametrar.

Uppstart av algoritmen kräver speciell hänsyn. Se diskussionen ovan. Första stegsvaret används alltså för att beräkna ett startvärde (Pmin, Jmin) till optimeringsalgoritmen. Först därefter kan sökningen starta. Detta gör att uppdateringen av Plow och Phigh fryses i det första samplingsintervallet och först i det andra löper uppdateringen enligt vad som nämnts i tidigare stycke. Initialt önskas inte uppdatering av något tillstånd. Vidare skall output ges initialvärdet av Pmin. Detta löses i båda fallen med villkorsatser.

Då få iterationer skall göras fås optimal startpunkt utifrån kvoten mellan två konsekutiva Fibonacci tal. Dessa beräknas genom att köra ett tidsdiskret system FIBONACI det antal steg som sökningen sedan skall göras. Först därefter kan procss, regulator och sökalgoritm sättas upp genom SYST kommandot. Lagring av initialvärdet från föregående simulering (med tidsdiskreta systemet Fibonaci) görs i en global INTRAC variabel genom kommandot: DISP p0/p0. Värdet av p0 överförs efter SYST kommadot till sökrutinen. Detta görs i ett macro: STARTUP.

I koden för själva sökalgoritmen ingår en del logiska villkor. Dessa uttryck förenklas avsevärt om den logiska operatorn: exklusivt-eller, införes. Då logiska värden representeras med 0 och 1 i simnon, kan exklusivt-eller enkelt erhållas genom modulo två addition. Modulo två räkning är en standard funktion i simnon.

En praktisk detalj har införts för att en regulator parameter som ger ett instabilt system inte skall ge problem. Problemet är helt av numerisk natur. Detta kringgås genom att SEARCH förutom att ge testpunkt Ptest också ger Jmax som utsignal. I processen jämförs hela tiden J med Jmax och överskrides Jmax avbrytes stegsvaret för detta k värde. Egentligen skulle Jmin kunna användas för detta ändamål, men det kan vara intressant att jämföra stegsvaren för de olika parametervärdena och därför har Jmax satts en god bit över Jmin.

Då stegsvaren för de olika parametervärdena presenteras med kommandot: show y(tau), fås en linje mellan ett stegsvars slutpunkt och nästa stegsvars startpunkt. Denna linje fyller ingen funktion utan förvirrar endast. Låter man istället plotta stegsvaren under simuleringens gång med kommandot plot y(tau) och med switchen DARK i läge ON undvikes detta.

Referenser

1) Elmqvist H, Åström K J, Schönthal T: SIMNON - user's guide for MS-DOS computers. Dept of Automatic Control, LTH, 1986.

2) Luenberger D: Linear and Nonlinear Programming. Kapitel 7.1. Addison and Wesley, 1984.

3) Wilde D J and Beightler C S: Foundations of Optimization. Prentice-Hall, Englewood Cliffs, N J, 1967.

```
DISCRETE SYSTEM search

"Linear search over the interval Plo ( P ( Phi for a minumum of J(P).
"The function J(P) is assumed to be unimodal.
"Ref: Linear and Nonlinear programming - D. Luenberger. Chap 7.1.

INPUT J
OUTPUT Ptest Jmax tbegin
STATE Plo  Phi  Pmin  P  Jmin
NEW  qPlo qPhi qPmin qP qJmin
TIME t
TSAMP ts

"Update the search state:
left    = IF (P(M)           THEN 1 ELSE 0
decr    = IF (J(Jmin)        THEN 1 ELSE 0
PloFix  = IF (t(1.5*length) THEN 1 ELSE MOD(left+decr+1,2)
PhiFix  = IF (t(1.5*length) THEN 1 ELSE MOD(left+decr,2)

U  =  newPhi-newPlo  "Magnitude of the uncertainity interval.
M  =   (Phi+Plo)/2   "Midpoint of the interval.

newPlo  = IF PloFix THEN Plo ELSE IF decr THEN Pmin ELSE P
newPhi  = IF PhiFix THEN Phi ELSE IF decr THEN Pmin ELSE P
newPmin = IF decr   THEN P   ELSE Pmin

qJmin   = IF (t)0.5*length) AND decr THEN J       ELSE Jmin
qPlo    = IF (t)1.5*length)          THEN newPlo  ELSE Plo
qPhi    = IF (t)1.5*length)          THEN newPhi  ELSE Phi
qPmin   = IF (t)0.5*length)          THEN newPmin ELSE Pmin
qP      = Ptest

"Calculate the new test point Ptest:
Ptest   = IF (t)0.5*length) THEN newPhi+newPlo-newPmin ELSE Pmin

"Stop the test of the regulator if J)Jmax!
Jmax    = IF (t)1.5*length) THEN 2*Jmin ELSE Jmax0

"Calculation of the start test point Ptest:
Pstart = Plo+g*(Phi-Plo)

"Zero process:
x1[process] = 0
x2[process] = 0
J[process]  = 0

"New sample:
ts     = t + length
tbegin = t

"Parameters:
length: 20

Jmin  : 1E10
Jmax0 : 10
Plo   : 0
Phi   : 1
Pmin  : 0.61803
g     : 0.61803 "Golden section ratio - exact value is 2/(1+sqrt(5)).

END
```

```
CONTINUOUS SYSTEM process

INPUT yr k Jmax tau
STATE x1  x2  J
DER  dx1 dx2 dJ

"Alarm:
alarm = IF J>Jmax THEN 1 ELSE 0

"Process:
dx1 = IF not alarm THEN -a*x1 + e ELSE 0
dx2 = IF not alarm THEN    x1     ELSE 0
y   = x2

"Regulator:
e   = k*(yr-y)

"Loss ITAE:
dJ = IF not alarm THEN tau*abs(e) ELSE 0

"Parameters:
a:1

END

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

CONNECTING SYSTEM tune

TIME t
yr[process]   = 1
J[search]     = J[process]
k[process]    = Ptest[search]
Jmax[process] = Jmax[search]
tau[process]  = t-tbegin[search]

END
```

```
MACRO startup testtime tunetime
let l.=testtime
let t.=tunetime
syst Fibonaci
let n.=t./l.
let n.=n.-1
simu 0 n.
disp p0/p0
setup
par g:p0.
par length:testtime
END
```

- -

```
DISCRETE SYSTEM Fibonaci

STATE F1  F2
NEW  qF1 qF2
TIME t
TSAMP ts

"Generation of the Fibonacci sequence:
qF2 = F1+F2
qF1 = F2
P0  = F1/F2
ts  = t+h

"Parameters:
h:1
F1:1
F2:1

END
```

- -

```
MACRO setup
syst process search tune
store y j[process] k newplo newphi pmin Jmin
END
```

- -

```
MACRO interval plow phigh
init plo:plow
init phi:phigh
simu 0 0.01
init pmin:pstart
init p:pstart
END
```

```
MACRO startup1 testtime tunetime plow phigh
free l.
free t.
free nmax.
free u.
let l.=testtime
let t.=tunetime
let nmax.=t./l.
quotient 0 nmax.
let u.=phigh-plow
let u.=invf1.*u.
let qgold.=0.618034
write '=================================='
write 'Total time for tests : 'tunetime
write 'i.e. number of tests : 'nmax.
write 'Final uncertainity   < 'u.
write 'Optimal ratio        : 'q.
write 'Golden ratio         : 'qgold.
write '=================================='
setup
par length:testtime
startP plow phigh q.
END
```

- -

```
MACRO startup2 testtime accuracy plow phigh
free nmax.
free normacc.
free uncert.
free tunetime.
let nmax.=50.
let qgold.=0.618034
let uncert.=phigh-plow
let normacc.=accuracy/uncert.
quotient normacc. nmax.
let tunetime.=n.*testtime
let u.=phigh-plow
let u.=invf1.*u.
write '========================================'
write 'Desired accuracy          : 'accuracy
write 'Required number of tests : 'n.
write 'i.e. total time           : 'tunetime.
write 'Optimal ratio             : 'q.
write 'Golden ratio              : 'qgold.
write '========================================'
setup
par length:testtime
startp plow phigh q.
END
```

```
MACRO quotient acc nmax
free f1.
free f2.
free qf1.
free qf2.
free q.
free invf1.
free n.
let f1.=1.
let f2.=1.
for i = 1. to nmax
let qf2.=f1.+f2.
let qf1.=f2.
let f1.=qf1.
let f2.=qf2.
let q.=f1./f2.
let invf1.=1./f1.
let n.=i
IF invf1. LE acc GOTO exit
next i
label exit
END
```

```
MACRO startP Plow Phigh q
"This macro assumes SETUP has been done.
free uncert.
free p0.
let uncert.=phigh-plow
let uncert.=q*uncert.
let p0.=plow+uncert.
par g:q
init plo:plow
init phi:phigh
init pmin:p0.
init p:p0.
end
```