

ISSN 0280-5316
ISRN: LUTFD2/TFRT--5480--SE

Olinjär Stokastisk Reglering av Extremvärden

Lennart Andersson

Institutionen for Reglerteknik
Lunds Tekniska Högskola
October 1993

Department of Automatic Control Lund Institute of Technology P.O. Box 118 S-221 00 Lund Sweden		<i>Document name</i> MASTER THESIS
		<i>Date of issue</i> September 1993
		<i>Document Number</i> ISRN LUTFD2/TFRT--5480--SE
<i>Author(s)</i> Lennart Andersson	<i>Supervisor</i> Anders Hansson, Karl Johan Åström	
	<i>Sponsoring organisation</i>	
<i>Title and subtitle</i> Non-Linear Stochastic Control of Extreme values, Olinjär Stokastisk Reglering av Extremvärden		
<i>Abstract</i> <p>In this report discrete time optimal control of extreme values are investigated. Both the full information case and partial information case are discussed. There has been payed special attention to numerical solution of the Bellman equation. For a special full information case the optimal controller has been obtained by a numerical solution. This controller has been compared with the minimal variance controller and the linear minimal upcrossing controller through simulations. It is seen that the optimal controller performs better than the linear controllers when the noise is high and the minimization horizon is short. Further, the optimal controller has been compared with the non-linear minimal upcrossing controller, which has been derived without using Bellman's equation. It is seen that the controllers were almost identical for the case studied, and for a special case it is shown that they are identical.</p>		
<i>Key words</i> Stochastic control, Extreme values, Optimal Control, Critical Processes		
<i>Classification system and/or index terms (if any)</i>		
<i>Supplementary bibliographical information</i>		
<i>ISSN and key title</i> 0280-5316		<i>ISBN</i>
<i>Language</i> Swedish	<i>Number of pages</i> 87	<i>Recipient's notes</i>
<i>Security classification</i>		

The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Fax +46 46 110019, Telex: 33248 lubbis lund.

Förord

Detta examensarbete har utförst som en sista del i min utbildning på elektrotekniklinjen vid Lunds Tekniska Högskola. Arbetet har utförts på institutionen för Reglerteknik under handledning av Anders Hansson och bygger på hans arbete Hansson (1993).

Under våren då jag skulle välja examensarbete blev jag snabbt övertygad om att jag ville göra något med inriktning mot stokastiska processer och helst också reglerteknik. Denna beslutsamhet finns att finna i de kurser jag läst under den senare delen av min utbildning. Speciellt kan nämnas Stokastiska processer, Tidsserie analys, Process identifiering och Adaptiv reglering på institutionerna för Matematisk statistik och Reglerteknik här i Lund.

På båda dessa institutioner fanns intressanta examensarbeten föreslagna liksom på en del företag. Mitt slutliga beslut tog jag när Per Hagander på Reglerteknik visade mig ett av Anders Hansson föreslaget examensarbete, Stokastisk reglering av extremvärdens. Jag vill därför tacka Per Hagander för att han ledde mig in på rätt examensarbete.

Under examensarbetets gång har flera personer varit till stor hjälp och inspiration. Jag kan inte nog tacka min handledare Anders Hansson för hans stora engagemang, tålmod och uppmuntran. För värdefull kunskap om numerisk lösning av Bellmans ekvation vill jag tacka professor Karl Johan Åström. Tack till Leif Andersson för praktisk hjälp med datorerna. Jag vill också tacka professor Georg Lindgren på institutionen för Matematisk statistik för de samtal som vi haft. Viktigt för arbetet har också den miljö i vilken jag jobbat varit. Därför vill jag tacka alla på Institutionen för Reglerteknik för deras glada, trevliga och hjälpsamma egenskaper.

Lund, september, efter en mycket solsnål sommar 1993
Lennart Andersson

Innehåll

Förord	1
1. Introduktion	5
Översikt	6
2. Problemformulering	7
2.1 Systemmodell	7
2.2 Styrkriterier	7
2.3 Sammanfattning	8
3. Dynamisk programmering	9
3.1 Generella steg vid optimeringen	9
3.2 Numerisk lösning	10
3.3 Sammanfattning	11
4. Fullständig tillståndsinformation	12
4.1 Det allmänna problemet	12
4.2 Numeriska beräkningar	12
4.3 Ett exempel	13
4.4 Ett fall med fullständig tillståndsinformation	14
4.5 Sammanfattning	21
5. Simuleringar	28
5.1 Approximativa regulatorer	28
5.2 Simuleringar	30
5.3 Minimalvariansregulator	34
5.4 Linjär minimaluppkorsningsregulator	36
5.5 Olinjär minimaluppkorsningsregulator	38
5.6 Sammanfattning	44
6. Ofullständig tillståndsinformation	47
6.1 Det allmänna problemet	47
6.2 Numeriska beräkningar	48
6.3 Simulering av ett filter	49
6.4 Sammanfattning	51
7. Slutsatser	55
Fortsatt arbete	55
Referenser	56
A. Ett bevis	58
B. Storheter och beteckningar	59
C. Programlistningar	60
Update.m	61
Integrate.c	63
InitK.m	67
GetU.c	68
GetV.c	71
Simu.m	74
intensityd.m	75
reguld.m	76
LQregul.m	78
Facit.c	79
Maxabsfilter.m	84
MakePlot.m	86

1. Introduktion

Denna rapport kommer att behandla reglering av extremvärden. Motiveringar till att studera reglering av extremvärden följer nedan och är hämtade från Hansson (1993) med fri översättning.

Många processer i industrin är kritiska. De är ofta kritiska i den mening att de har en begränsande nivå. Denna kan vara antingen fysisk eller artificiell. Exempel på det förra är sådana nivåer, som inte kan överskridas utan katastrofala följer, t.ex. explosioner. Ett exempel på det senare är alarmnivåer, som om de överskrids framtvingar nödstopp eller förändringar i driften. Ett annat exempel är kvalitetsnivåer, som om de överskrids leder till missnöjda kunder. Gemensamt för de kritiska processerna är att de träder in i den kritiska regionen abrupt i och med, att en signal överskrider en begränsande nivå.

Avståndet mellan den begränsande eller kritiska nivån och referensvärdet är normalt stort, eftersom antalet gränsöverskridanden för den reglerade nivån annars skulle bli oacceptabelt många. Å andra sidan kan det finnas andra regleringsmål, som gör det öönskat eller omöjligt att välja avståndet stort. Ett exempel på problem av denna typ finns beskrivet i Borisson och Syding (1976), där effekten till en malmkross skall hållas så hög som möjligt utan att överskrida en viss nivå för att undvika att överlastskyddet löser ut. Ett annat exempel är fuktighetsreglering i en pappersmaskin, där det är önskvärt att hålla fukthalten så hög som möjligt utan att orsaka våta ränder, Åström (1970). Ännu ett problem är kraftreglering i vindkraftverk, där det övervakande systemet kopplar ner, om den genererade effekten överskrider 140 % av den nominella effekten, Mattsson (1984). Andra exempel kan hittas i sensorbaserad robotstyrning och kraftreglering, Hansson och Nielsen (1991), och reglering av olinjära system, där stabiliteten kan bli tillståndsberoende, Shinskey (1967).

Deterministiska problem av denna typ kan lösas genom att minimeras:

$$\max_d ||z||_\infty$$

där z är den reglerade signalen och d störningar som påverkar z. Problem av denna typ har studerats utförligt. Under antagandet att processen är linjär och störningar har begränsad energi fås den välkända H_2 -regulatorn, Vidyasagar (1986). Andra typer av störningar har också studerats. I Vidyasagar (1986) och Dahleh och Pearson (1987) har störningarna begränsad supremumnorm, och i Liu och Zakian (1990) har de begränsade ökningar.

Gemensamt för de deterministiska kriterierna är att designen sker för värsta fallet av störning, vilket kan tyckas något konserverativt. Det klassiska sättet att komma förbi detta är en stokastisk problemformulering. Detta finns beskrivet i Hansson (1992), där approximativa lösningar genom den s.k. minimaluppkörsningsregulatorn (MU) har erhållits för kriteriet

$$P \left\{ \max_{0 \leq k \leq N} z(k) > z_0 \right\} \quad (1.1)$$

där z_0 är avståndet till den kritiska nivån. Detta kriterium kan också

approximativt minimeras genom minimalvariansregulatorn (MV), se Åström (1970), Åström och Wittenmark (1990) och Borisson och Syding (1976). Förstärkningen hos minimalvariansregulatorn beror kritiskt på samplingsperioden. För kort samplingsperiod leder till stora variationer i styrsignal., Åström och Wittenmark (1990). Detta problem har lösats genom att införa viktning av styrsignalen—LQG-design. Emellertid har man inte hittat något bra kriterium för valet av viktning. MU regulatorn kan tolkas som att välja optimal viktning i ett LQG-problem. I Hansson (1992) jämförs MU-regulatorn och MV-regulatorn med avseende på ovanstående kriterium. Det visar sig att det finns exempel där MU-regulatorn har upp till 10 % bättre uppförande.

Tidigare har endast linjära regulatorer behandlats. I denna rapport skall sådana olinjära regulatorer studeras, som minimerar ekvation 1.1. Problemet kan omformuleras genom dynamisk programmering vilket resulterar i Bellmans ekvation, se Åström (1977). Denna ekvation beskriver ett sätt att rekursivt finna den optimala styrlagen. Ekvationen är oftast omöjlig att lösa analytiskt, varför man blir hänvisad till numerisk lösning. Principerna för hur detta skall gå till diskuteras ingående både vid fullständig och ofullständig tillståndsinformation. Vid fullständig tillståndsinformation löser vi ett exempel numeriskt och jämför med minimalvarians- och minimaluppkorsningsregulatorerna nämnda ovan. För ett speciellt fall av exemplet härleds den optimala regulatorn utan att lösa Bellmans ekvation. I det fallet blir den optimala regulatorn ekvivalent med den s.k. olinjära minimaluppkorsningsregulatorn.

Översikt

Denna rapport är upplagd enligt följande. I kap 2 ges problemformuleringen. Systemet beskrivas av linjära differensekvationer. Regulatorn erhålls genom att minimera en förlustfunktion som tillåts bero av ett maximum relaterat till processen. I kap 3 visas hur man kan använda dynamisk programmering för att lösa optimeringsproblemet. Detta resulterar i Bellmans ekvation. Vid fullständig tillståndsinformation kan vissa förenklingar göras. Detta beskrivs i kap 4. Där presenteras också en numerisk lösning av Bellmans ekvation för ett andra ordningens system och styrkriteriet (1.1). I kap 5 görs simuleringar av den i kap 4 numeriskt erhållna regulatorn. Jämförelser görs med ett antal approximativt optimala regulatorer. Problemet med ofullständig tillståndsinformation dissuteras i kap 6. I kap 7 ges slutsatser och ideér om fortsatt arbete.

2. Problemformulerings

I detta kapitel införs systembeskrivningen och styrkriteriet. Systemet ges av en allmän linjär och tidsinvariant differensekvation. Ett systemrelaterat maximum införs som ett extra tillstånd att använda i styrkriteriet. Styrkriteriet definieras så att det förutom det olinjära maximumet också tillåts bero av styrsignaler, tidpunkter och det linjära systemets tillstånd.

2.1 Systemmodell

Låt systemet beskrivas av följande linjära tidsdiskreta system

$$\begin{cases} x(k+1) &= \Phi x(k) + \Gamma u(k) + v(k) \\ y(k) &= C_1 x(k) + e(k) \\ z(k) &= C_2 x(k) \end{cases}$$

Storheten x är systemets tillstånd, y dess utsignal, z den reglerade storheten, u är styrsignalen och v samt e är processbrus respektive mätbrus. Brusen antags vara vita och normalfördelade med väntevärde noll. Ingen korrelation mellan v och e förekommer. Som beteckningar för brusens varianser införs $R_1 = Evv^T$ och $R_2 = Eee^T$. Dessutom gäller att $x(0)$ är en stokastisk storhet med väntevärde m_0 och kovarians R_0 . Sammantaget gäller alltså

$$\begin{cases} v(k) \in N(0, R_1) & \text{vitt brus} \\ e(k) \in N(0, R_2) & \text{vitt brus} \\ x(k) \in N(m_0, R_0) \end{cases}$$

Som en storhet att använda i styrkriteriet införs ξ som det största värdet på $g(z)$ som uppnåtts sedan start. Funktionen g är en växande reellvärd funktion med $g(0) = 0$. Matematiskt kan vi teckna ξ genom en rekursion

$$\xi(k+1) = \max[\xi(k), g(z(k+1))]$$

med initialvärdet $\xi(0) = g(z(0))$. I fortsättningen kallas $[x^T(k) \ \xi(k)]^T$ det utökade tillståndet medan $x(k)$ kallas tillståndet.

2.2 Styrkriterier

Som regleringens främsta uppgift väljs att maximumet inte skall bli för stort. Förutom detta skall det vara möjligt att införa önskemål på att styrsignalen hålls liten och/eller att tillståndet inte avviker från sitt konstanta referensvärdet i onödan. Det konstanta referensvärdet kan väljas till noll utan någon inskränkning av problemställningen. Detta kommer att utnyttjas i fortsättningen.

Den önskade regleringsstrategin kan matematiskt beskrivas som att följande förlustfunktion skall minimeras

$$J(p(x(0), \xi(0)), u(\cdot)) = E \left\{ \sum_{k=0}^N h(k, x(k), \xi(k), u(k)) \right\}$$

Det gäller att h är en reellvärd funktion av ett antal variabler. Dessa variabler kan ”straffas” olika genom valet av funktionen h . Man kan t.ex. låta bli att straffa vissa variabler om man så önskar. I de problem som här kommer att studeras gäller generellt att maximumet ξ alltid straffas. Detta sker alltid på ett sådant sätt att straffet växer, inte nödvändigtvis kontinuerligt, med ξ . Om maximumet inte straffas blir allt utom styrkriteriet linjärt vilket förenklar problemställningen avsevärt. Detta kommer inte att studeras.

Styrkriteriet visar att det gäller att minimera förlusten under $N+1$ sampel och inte över en oändlig tid. Det skall noteras att J är den förväntade förlusten och inte den verkliga. Notera också att förlustfunktionen beror av den initiala fördelningen och av de styrsignaler som läggs på. I fortsättningen kallas J förlusten.

2.3 Sammanfattning

I detta kapitel har systemet införts tillsammans med ett styrkriterium. Systemet valdes linjärt och tidsinvariant i diskret tid. Ett systemrelaterat maximum infördes. Styrkriteriet definierades på ett sådant sätt att det tilläts bero av tidpunkt, tillstånd, styrsignal och maximum.

3. Dynamisk programmering

I detta kapitel beskrivs hur man med hjälp av dynamisk programmering kan lösa optimeringsproblemet från föregående kapitel. Detta leder i avsnittet 3.1 fram till den s.k. Bellmans ekvation. Eftersom denna ekvation är en funktionalekvation, kan den normalt sett inte lösas analytiskt. I avsnitt 3.2 diskuteras därför numeriska lösningsmetoder. Dessa kommer att tillämpas på ett exempel i kapitel 4.

3.1 Generella steg vid optimeringen

Förlustfunktionen betecknas med J och representerar förlusten under $N + 1$ sampel. Denna storhet skall minimeras av styrlagen. Det bidrag till den minimala förlusten som förväntas uppstå från sampel k fram till det sista samplet N då informationen $\mathcal{Y}(k)$ är given definieras som $V(k)$. Denna storhet införs för optimeringen och definieras som

$$V(k, p(x(k), \xi(k)|\mathcal{Y}(k))) = \min_{u(i): k \leq i \leq N} E \left\{ \sum_{i=k}^N h(i, x(i), \xi(i), u(i)) | \mathcal{Y}(k) \right\}$$

För att hitta ett samband mellan V och optimeringsproblemet skall först den tillgängliga informationen vid tidpunkten noll, $\mathcal{Y}(0)$, studeras. Enligt problemställningen är den initiala tätheten för tillståndet alltid given. Den extra information som kan uppnås vid tiden noll måste därför komma från mätningen vid tiden noll. Denna mätning blir en stokastisk storhet i det fall mätbrus existerar. Sammanfattningsvis gäller att $\mathcal{Y}(0)$ kan vara en stokastisk storhet. Detta ger följande samband

$$E \{ V(0, p(x(0), \xi(0)|\mathcal{Y}(0))) \} = \min_{u(k): 0 \leq k \leq N} J(p(x(0), \xi(0)), u(\cdot))$$

Detta innebär att förväntat V vid tidpunkten noll ger den minimala förlust som kan uppnås. Den styrlag som leder till detta värde är den optimala styrlagen. Optimering enligt dynamisk programmering ger nu följande samband kallat Bellmans ekvation

$$\begin{aligned} V(k, p(x(k), \xi(k)|\mathcal{Y}(k))) &= \min_{u(k)} E \{ h(k, x(k), \xi(k), u(k)) \\ &\quad + V(k+1, p(x(k+1), \xi(k+1)|\mathcal{Y}(k+1))) | \mathcal{Y}(k) \} \end{aligned}$$

för $0 \leq k \leq N - 1$ med slutvärde

$$V(N, p(x(N), \xi(N)|\mathcal{Y}(N))) = \min_{u(N)} E \{ h(N, x(N), \xi(N), u(N)) | \mathcal{Y}(N) \}$$

Bellmans ekvation ger ett rekursivt samband för att lösa vårt problem. Den dynamiska programmeringen grundar sig på att förlustbidraget h vid en viss tidpunkt inte påverkas av styrsignaler efter denna tidpunkt.

Observera hur den givna informationen kommer in. När vi skall bestämma $u(k)$ finns den information som får användas $\mathcal{Y}(k)$ given. Minimering över en styrsignal åt gången med tillåten information given visar anledningen till att införa V vid optimeringen.

Den givna informationen beror av problemställningen. I fallet med fullständig tillståndsinformation är det utökade tillstånden givet fram till aktuell tidpunkt. Vid ofullständig tillståndsinformation är uppmätta utsignaler givna.

Normalt sett är det inte möjligt att lösa Bellmans ekvation analytiskt. Detta leder till att man måste göra numeriska beräkningar.

3.2 Numerisk lösning

Bellmans ekvation ges av

$$V(k, p(x(k), \xi(k)|\mathcal{Y}(k))) = \min_{u(k)} E\{h(k, x(k), \xi(k), u(k)) + V(k+1, p(x(k+1), \xi(k+1)|\mathcal{Y}(k+1)))|\mathcal{Y}(k)\}$$

för $0 \leq k \leq N - 1$ med slutvärde

$$V(N, p(x(N), \xi(N)|\mathcal{Y}(N))) = \min_{u(N)} E\{h(N, x(N), \xi(N), u(N))|\mathcal{Y}(N)\}$$

För de flesta problemen måste numeriska lösningsmetoder tas till för att lösa Bellmans ekvation. Denna lösning kan delas in i ett antal delproblem som beskrivs nedan. Beroende på det specifika problemet kan oftast vissa av dessa delproblem reduceras i både svårighet och storlek.

Lagring av minimala delförlusten V i en tabell Vid den numeriska beräkningen måste den minimala delförlustfunktionen

$$V(k+1, p(x(k+1), \xi(k+1)|\mathcal{Y}(k+1)))$$

finnas lagrad för alla möjliga täthetsfunktioner eftersom $V(k+1)$ behövs vid bestämningen av $V(k)$ och $u(k)$. Tätheterna för vilka $V(k+1)$ måste vara lagrad bestäms av alla möjliga uppsättningar på $\mathcal{Y}(k+1)$. Dessa är många

I fallet fullständig tillståndsinformation beror alla möjliga täthetsfunktioner bara av det utökade tillståndet eftersom detta representerar hela den givna informationen. Därför räcker det att lagra V för alla värden på det utökade tillståndet vid varje tidpunkt. Detta är inte praktiskt möjligt utan kräver en diskretisering av de utökade tillståndens värden. Valet av diskretisierungspunkter är kritiskt och måste väljas med omsorg. Symmetri i V reducerar ofta antalet punkter som behöver lagras. Värdet i diskreta punkter kan sedan tillämpas med interpolation och extrapolation användas som en approximation av V .

När man inte har fullständig tillståndsinformation blir det något mer komplicerat och detta kommer att diskuteras mera ingående i kapitel 6.

Väntevärdesbestämning Initieringen av tabellen med V -värden använder följande väntevärdesbestämning

$$E\{h(N, x(N), \xi(N), u(N))|\mathcal{Y}(N)\}$$

Uppdateringen av tabellen utnyttjar att man har tillgång till $V(k+1)$ för varje tänkbar täthetsfunktion $p(x(k+1), \xi(k+1)|\mathcal{Y}(k+1))$ då väntevärdet nedan skall bestämmas

$$E\{h(k, x(k), \xi(k), u(k)) + V(k+1, p(x(k+1), \xi(k+1)|\mathcal{Y}(k+1)))|\mathcal{Y}(k)\}$$

Denna väntevärdesbildning sker över det utökade tillståndet vars täthet $p(x(k), \xi(k)|\mathcal{Y}(k))$ är en inparameter. Problemet att m.h.a. denna täthet beräkna $p(x(k+1), \xi(k+1)|\mathcal{Y}(k+1))$ kallas filtreringsproblemet och kommer att diskuteras senare.

Minimering När de tidigare stegen är färdigimplementerade så skall man utföra ovanstående väntevärdesbildning för alla möjliga värden på $u(k)$ och sedan välja det $u(k)$ som ger lägst väntevärde. Detta är inte praktiskt möjligt utan räkningarna kan bara utföras för ett begränsat antal värden på $u(k)$. Detta kan till viss del kompenseras av interpolationer. Vid interpoleringen fås förhoppningsvis ett bra värde på det optimala $u(k)$. För detta värde kan man sedan beräkna väntevärdet en sista gång och få fram ett förbättrat värde på V .

Updatering Genom att upprepa ovanstående steg kommer nya tabeller med V att växa fram. När dessa bestäms erhålls samtidigt värden på de önskade optimala styrsignalerna.

3.3 Sammanfattning

I detta kapitel har beskrivits hur man med hjälp av dynamisk programmering kan omformulera problemställningen i kapitel 2 rekursivt. Den ekvation som då erhålls kallas Bellmans ekvation. Hur denna kan lösas numeriskt diskuterades i slutet av kapitlet.

4. Fullständig tillståndsinformation

I detta kapitel behandlas fallet med fullständig tillståndsinformation. Detta innebär att tillståndet är känt vid den tidpunkt då styrsignalen skall bestämmas. Detta innebär avsevärda förenklingar.

I avsnitt 4.1 studeras hur fullständig tillståndsinformation förenklar Bellmans ekvation. Dess numeriska lösning diskuteras i avsnitt 4.2. Ett exempel införs i avsnitt 4.3 och en analytisk lösning presenteras. Exemplet saknar analytisk lösning i vissa fall. Ett sådant fall studeras i avsnitt 4.4 där en numerisk lösning av Bellmans ekvation utförs.

4.1 Det allmänna problemet

Vid fallet fullständig tillståndsinformation gäller att den givna informationen är

$$\mathcal{Y}(k) = \{x(i) : 0 \leq i \leq k\}$$

Detta innebär att $x(k), \xi(k)|\mathcal{Y}(k)$ helt enkelt blir $x(k), \xi(k)$ vilket medför att Bellmans ekvation kan skrivas som:

$$\begin{aligned} V(k, x(k), \xi(k)) &= \min_{u(k)} E\{h(k, x(k), \xi(k), u(k)) \\ &\quad + V(k+1, x(k+1), \xi(k+1))|x(k), \xi(k)\} \\ &= \min_{u(k)} [h(k, x(k), \xi(k), u(k)) \\ &\quad + E\{V(k+1, x(k+1), \xi(k+1))|x(k), \xi(k)\}] \end{aligned}$$

för $0 \leq k \leq N-1$ med slutvärdet

$$V(N, x(N), \xi(N)) = \min_{u(N)} h(N, x(N), \xi(N), u(N))$$

Observera att minimala delförlustfunktionen V nu inte beror av en täthetsfunktion utan bara av det utökade tillståndet $[x^T(k) \ \xi(k)]^T$ och tidpunkten.

4.2 Numeriska beräkningar

Det faktum att V bara beror av det utökade tillståndet innebär för de numeriska beräkningarna att V bara behöver lagras för alla möjliga tillstånd istället för att lagras för alla möjliga täthetsfunktioner. Som tidigare beskrivits bör naturligtvis eventuella symmetrier utnyttjas för att reducera lagringstallerna.

Väntevärdesbestämningen ges av följande integral

$$\begin{aligned} E\{V(k+1, x(k+1), \xi(k+1))|x(k), \xi(k)\} &= \\ \int p(x(k+1), \xi(k+1)|x(k), \xi(k)) \\ \cdot V(k+1, x(k+1), \xi(k+1)) dx(k+1) d\xi(k+1) \end{aligned}$$

Denna integration har samma dimension som x och ξ tillsammans. Täthetsfunktionen vid integrationen kan bestämmas analytiskt. Detta inses genom att det utökade tillståndet är givet och att bruset är normalfördelat.

4.3 Ett exempel

I detta avsnitt skall vi studera ett exempel hämtat från Hansson (1993).

Modell och styrkriterium

Låt det linjära systemet beskrivas av den allmänna modellen

$$\begin{cases} x(k+1) &= \Phi x(k) + \Gamma u(k) + v(k) \\ y(k) &= C_1 x(k) + e(k) \\ z(k) &= C_2 x(k) \end{cases}$$

Vi väljer att maximumet ξ skall representera det maximala absolutbeloppet på z

$$\xi(k+1) = \max(\xi(k), |z(k)|)$$

Styrkriteriet låter vi definieras som att minimera förlustfunktionen

$$J(u(\cdot)) = P\{\xi(N) \geq \xi_0\}$$

Dessa speciella val av maximum och styrkriterium kan i den allmänna beskrivningen införas genom att välja

$$g(z) = |z|$$

respektive

$$h(k, \xi(k)) = \begin{cases} 0 & , 0 \leq k \leq N-1 \\ \theta(\xi(N) - \xi_0) & , k = N \end{cases}$$

där θ är stegfunktionen definierad av

$$\theta(z) = \begin{cases} 0 & z \leq 0 \\ 1 & z > 0 \end{cases}$$

Bellmans ekvation

Tidigare visades att den optimala lösningen ges av Bellmans ekvation. I exemplet vi studerar blir den

$$V(k, x(k), \xi(k)) = \min_{u(k)} E\{V(k+1, x(k+1), \xi(k+1)) | x(k), \xi(k)\}$$

för $0 \leq k \leq N-1$ med slutvärdet

$$V(N, x(N), \xi(N)) = \min_{u(N)} E\{h(N, \xi(N))\} = \theta(\xi(N) - \xi_0)$$

Den sista likheten följer av att styrsignalen inte hinner påverka maximumet. Väntevärdesbildningen har tidigare tagits fram

$$\begin{aligned} E\{V(k+1, x(k+1), \xi(k+1)) | x(k), \xi(k)\} &= \\ &\int p(x(k+1), \xi(k+1) | x(k), \xi(k)) \\ &\cdot V(k+1, x(k+1), \xi(k+1)) dx(k+1) d\xi(k+1) \end{aligned}$$

Analytisk lösning

Antag att $V(k+1)$ bara beror av $\xi(k+1)$ och $k+1$. Då kan Bellmans ekvation skrivas som

$$V(k, \mathbf{x}(k), \xi(k)) = \min_{u(k)} \int p(\xi(k+1)|\mathbf{x}(k), \xi(k)) \\ \cdot V(k+1, \xi(k+1)) d\xi(k+1)$$

Tätheten ovan får enligt Hansson (1993) följande utseende

$$p(\xi(k+1)|\mathbf{x}(k), \xi(k)) = \frac{1}{\sigma} \varphi \left(\frac{\xi(k+1) - m(k)}{\sigma} \right) + \frac{1}{\sigma} \varphi \left(\frac{\xi(k+1) + m(k)}{\sigma} \right)$$

då $\xi(k) \leq \xi(k+1)$ och noll för övrigt. Ovan har införts följande beteckningar

$$m(k) = C_2(\Phi \mathbf{x}(k) + \Gamma u(k)) \\ \sigma^2 = C_2 R_1 C_2^T$$

där φ är den standardiserade normalfördelningens täthetsfunktion. Enligt Hansson (1993) går det att visa att det optimala valet av $u(k)$ är det som löser ekvationen $m(k) = 0$ om den har en lösning. I SISO-fallet gäller att lösning till $m(k) = 0$ existerar precis då $C_2 \Gamma \neq 0$. Med denna lösning insatt ser vi att $\mathbf{x}(k)$ elimineras ur tätheten så att $V(k, \mathbf{x}(k), \xi(k))$ inte beror av $\mathbf{x}(k)$. Eftersom $V(N, \mathbf{x}(N), \xi(N))$ inte är en funktion av $\mathbf{x}(N)$ följer av induktionsprincipen att $p(k, \mathbf{x}(k), \xi(k))$ är oberoende av $\mathbf{x}(k)$ för $0 \leq k \leq N$. Lösningen ges då av

$$u(k) = -\frac{C_2 \Phi}{C_2 \Gamma} \mathbf{x}(k)$$

Detta är minimalvarianslösningen m.a.p. $\mathbf{z}(k)$. Om $C_2 \Gamma = 0$ blir $m(k) \neq 0$ oavsett styrsignalens värde. För detta fall gäller att V inte blir oberoende av $\mathbf{x}(k)$. Lösningen för detta fall har inte studerats närmare i Hansson (1993). Ett sådant fall kommer att lösas numeriskt i nästa avsnitt.

4.4 Ett fall med fullständig tillståndsinformation

I föregående avsnitt studerades ett exempel där man önskade att maximumt med minsta möjliga sannolikhet skulle överskrida en gräns efter N sampel. Detta exempel kunde lösas analytiskt då $C_2 \Gamma \neq 0$. I detta avsnitt skall vi studera hur man för ett exempel med $C_2 \Gamma = 0$ numeriskt kan lösa problemet.

Val av system

I exemplet i föregående avsnitt hämtat från Hansson (1993) fick vi veta att lösningen blir minimal varians om $C_2 \Gamma \neq 0$. Villkoret $C_2 \Gamma = 0$ leder till att

$$\mathbf{z}(k+1) = C_2 \Phi \mathbf{x}(k) + v(k)$$

vilket innebär att tidsfördröjningen från styrsignal till $\mathbf{z}(k) = C_2 \mathbf{x}(k)$ måste vara minst två. Detta är ett nödvändigt men inte tillräckligt krav för att få en lösning skild från minimal varians.

Enligt experiment verkar det nödvändigt att införa färgat brus för att den optimala regulatorn inte skall bli lika med minimalvariansregulatorn.

Modell och styrkriterium

Samma modell och styrkriterium som i exemplet i föregående avsnitt skall studeras för ett speciellt val av andra ordningens system och maximum. Låt systemet vara en dubbelintegrator med korrelerat brus och med ett maximum som ges av största värdet på $|x_1|$. Detta kan uppnås genom att välja systemet som

$$\begin{cases} x_1(k+1) &= x_1(k) + x_2(k) + c w(k) \\ x_2(k+1) &= x_2(k) + w(k) + u(k) \end{cases} \quad (4.2)$$

där bruset w har standardavvikelse σ_w . Detta val av system gör att fördöjningen blir två sampel och att bruset blir färgat, åtminstone då $c \neq 0$.

I exemplet motsvarar detta att vi valt bruset som

$$v(k) = \begin{pmatrix} c \\ 1 \end{pmatrix} w(k)$$

och matriser enligt

$$\Phi = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad \Gamma = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad R_1 = \sigma_w^2 \begin{pmatrix} c^2 & c \\ c & 1 \end{pmatrix} \quad C_2 = (1 \ 0)$$

Initialtillståndet är slumpmässigt men mätbart. Storheten R_0 är därför ointressant.

Steg mot lösningen

När nu problemet skall lösas finns mycket att vinna genom att göra förenklingar. Dessa förenklingar reducerar ofta problemets storlek och svårighetsgrad. Reducerad svårighet förenklar implementeringen. Minskad problemstorlek kommer främst in som förkortade exekveringstider vilket är av betydelse i detta sammanhang.

Bellmans ekvation har beskrivits i exemplet i föregående avsnitt och ges av

$$V(k, x(k), \xi(k)) = \min_{u(k)} E\{V(k+1, x(k+1), \xi(k+1)) | x(k), \xi(k)\}$$

för $0 \leq k \leq N-1$ med slutvärdet

$$V(N, x(N), \xi(N)) = \min_{u(N)} E\{h(N, \xi(N))\} = \theta(\xi(N) - \xi_0) \quad (4.3)$$

Väntevärdesbestämningen kan uttryckas som

$$\begin{aligned} E\{V(k+1, x(k+1), \xi(k+1)) | x(k), \xi(k)\} &= \\ &\int p(x(k+1), \xi(k+1) | x(k), \xi(k)) \\ &\cdot V(k+1, x(k+1), \xi(k+1)) dx(k+1) d\xi(k+1) \end{aligned}$$

Integrationen skall alltså utföras i tre dimensioner eftersom vi har två tillstånd i systemmodellen. Genom att utnyttja tolkningen av integrationen som väntevärde av delförlusten V givet det utökade tillståndet $[x^T(k) \ \xi(k)]^T$ kan problemet förenklas.

Genom att titta på systemekvationen ser man att den enda stokastiska storheten är bruset $w(k)$. Värdet på $w(k)$ och de givna tillstånden i kombination med en viss styrsignal bestämmer entydigt tillståndet vid nästa tidpunkt. Genom att väntevärdesbilda över $w(k)$ följer att trippelintegralen kan ersättas av följande enkelintegral

$$E\{V(k+1, x(k+1), \xi(k+1))|x(k), \xi(k)\} = \\ \int p(w) \cdot V(k+1, x(k+1), \xi(k+1)) dw \quad (4.4)$$

Eftersom w är normalfördelad är täthetsfunktionen given och de problem som kvarstår att lösa är

- val av implementeringsspråk
- för vilka tillstånd skall V lagras och u bestämmas
- interpolation och extrapolation i tabellen över V
- numerisk integration och minimering av V
- administrativa MATLAB rutiner

Först beskrivs de valda implementeringsspråken och därefter studeras övriga fyra punkter.

Implementeringsspråk

Implementeringen har skett i MATLAB 4.1 med vissa specialskrivna rutiner i C. MATLAB 4.1 är lämpligt för matrisräkning och grafisk presentation. Dock är det så att MATLAB's for-loopar och if-villkor inte är speciellt snabba varför programmen blir onödigt långsamma då man utnyttjar sådana konstruktioner.

Ett sätt att bibehålla fördelarna med MATLAB och samtidigt få snabba program uppnås genom att utnyttja något som kallas MEX-filer. Detta är ett sätt att göra C-filer körbara från MATLAB. Man utnyttjar därvid ett speciellt gränssnitt till MATLAB samtidigt som man utnyttjar delar av den matriss-truktur som MATLAB använder internt. MEX-filerna, d.v.s. C-rutinerna har utnyttjats för de beräkningar som sker på lägsta nivå. På så sätt har integrationerna vid väntevärdesbildningarna snabbats upp.

MATLAB MATLAB är ett interaktivt program som på ett effektivt sätt utför matrisoperationer och grafiska presentationer. Det går att skriva samman en mängd MATLAB kommandon i separata filer som sedan kan utnyttjas genom att ange filens namn. På detta sätt kan även egna funktioner implementeras. Namnet MATLAB står för MATrix LABratory och beskrivs i boken MATLAB (1992b).

MEX filer Eftersom MATLAB ursprungligen är gjort för interaktiv användning och inte för att skriva program blir beräkningar onödigt långsamma då man använder for-loopar och if-satser. Detta problem kan till stor del undvikas om de tidsödande rutinernas centrala delar skrivs om till C-program (eller Fort). Detta är möjligt genom att man skriver egna C-moduler. Modulerna skall vara gjorda enligt ett speciellt mönster så att de passar ihop med MATLAB. Dessa kompileras sedan av en rutin cmex till en användbar MATLAB-rutin. Dessa fungerar som vanliga MATLAB-funktioner (m-filer) vad gäller anrop. MEX beskrivs i boken MATLAB (1992a).

Vid implementeringen av C-modulerna gäller att väsentliga delar av de typer som specificerar matriserna är öppna. Detta gör att man effektivt kan

hantera matriserna i sina C-program. Dessutom finns färdiga rutiner att anropas från C-modulerna som t.ex. felmeddelanderutiner om antalet parametrar eller dimensioner inte stämmer. Programspråket C beskrivs i Kernighan och Ritchie (1978).

Metoder och Datastrukturer vid implementeringen

Nedan beskrivs själva implementeringen. Delar av metoderna har inspirerats av Helmersson (1981) och Åström och Helmersson (1986).

Att lagra minimala delförlusten V i en tabell Tabellen för den minimala delförlusten V måste lagras för ett antal diskreta värden på tillståndet. Valet av punkter är kritiskt och måste ske med omsorg. Hur valet gjorts kommer att diskuteras här.

Först skall diskuteras vilka ξ -värden som tabellen måste innehålla. I appendix visas att V antar samma värde för alla $\xi < \xi_0$. För ξ större än ξ_0 vet vi att V , oavsett andra tillstånd, blir 1. Detta innebär att V är känd då $\xi > \xi_0$ och inte behöver lagras för dessa ξ -värden. Då $\xi < \xi_0$ räcker det att lagra V för ett ξ -värde.

När det gäller x_1 vet vi att dess maximala absolutbelopp är detsamma som ξ varför det inte är någon mening att lagra V för $|x_1| > \xi_0$ ty då är $V = 1$.

För x_2 finns inga liknande begränsningar men symmetrin i V kan också utnyttjas! Genom att substituera x med $-x$ i systemet följer av förlustfunktionens och systemets symmetri att

$$V(-x_1, -x_2) = V(x_1, x_2)$$

Detta innebär att vi kan dela x_1, x_2 -planet med en rät linje genom origo och att vi bara behöver lagra V för ena halvan. Det kvarstår att bestämma hur denna räta linje skall väljas. Detta val blir uppenbart om vi först studerar vilken diskretisering av x_1 och x_2 som skall väljas.

Valet av diskretisering skall ske på ett sådant sätt att interpolationen blir bra. Detta innebär att de diskreta punkterna skall ligga tätt där V varierar snabbt. Vi vill nu uppskatta var detta sker. Genom att titta på systemmodellen ser vi att tillståndet x_1 vid nästa tidpunkt är summan av de båda tillstånden vid föregående tidpunkt, särskilt som på en brusterm. Eftersom x_1 bestämmer ξ och därmed förlusten verkar det enligt systemekvation (4.2) vara så att linjen $x_1 + x_2 = \xi_0$ är en brytgång för V . Nära denna linje bör diskretiseringspunkterna ligga tätt. På lite större avstånd från densamma kan man diskretisera glesare. På samma sätt är gränslinjen $|x_1| = \xi_0$ en brytgräns. Skillnaden är att denna gräns är abrupt och inte kontinuerlig m.a.p. V . Kommer man utanför är $V = 1$ och inte bara nära ett. Detta gör att det nära denna gräns inte nödvändigtvis måste vara tätare mellan diskretiseringspunkterna.

Punkternas täthet skall bero av avståndet till gränslinjen $x_1 + x_2 = \xi_0$. Ett sådant beroende är enklare att införa efter följande variabeltransformation

$$\begin{cases} s &= x_1 \\ t &= x_1 + x_2 \end{cases}$$

Genom denna transformation blir också valet av begränsningslinje enkelt. Välj $t = 0$. Den halva av planet som V skall lagras för väljs till $t \geq 0$.

Till sist kvarstår själva punktvalet. För endamålet införs normalerade storheter s_N och t_N . Dessa ligger båda strängt innanför gränserna plus och minus

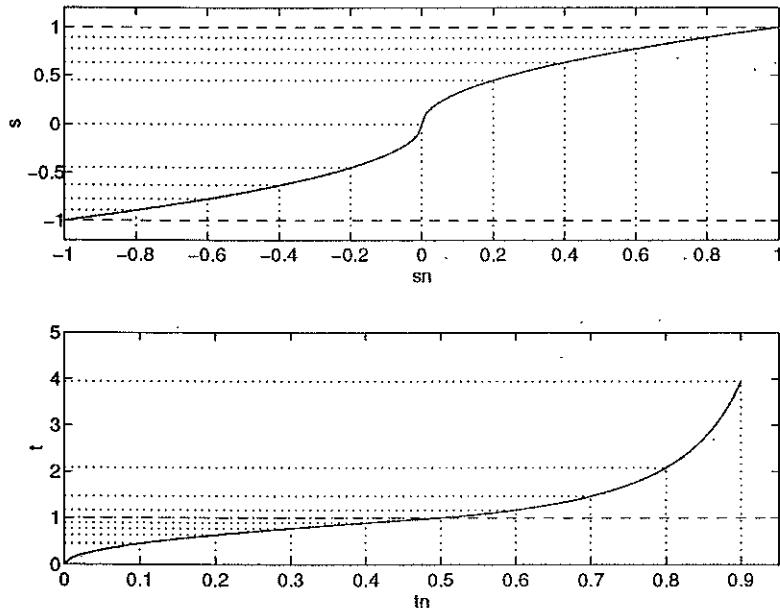
ett. Inuti tabellen skall dock inte V motsvarande negativa t_N lagras. Transformationen från normerade storheter väljs enligt

$$\begin{cases} s &= \xi_0 \sqrt{|s_N|} \operatorname{sign}(s_N) & -1 < s_N < 1 \\ t &= \begin{cases} \xi_0 \sqrt{2t_N} & 0 \leq t_N < 0.5 \\ \xi_0 \frac{0.75}{1-t_N^2} & 0.5 \leq t_N < 1 \end{cases} \end{cases}$$

Värdena på de normerade storheterna för vilka V skall lagras väljs linjärt i sitt område så att dessa enkelt kan kopplas till ett heltalet motsvarande en position i tabellen. Transformerna finns åskådliggjorda i figur 4.1 för det fall $\xi_0 = 1$. Punktlinjerna är till för att tydliggöra att transformen väljer punkterna tätare vid de streckade gränslinjerna. De inversa transformererna blir

$$\begin{cases} s_N &= \operatorname{sign}(s) \cdot \left(\frac{s}{\xi_0}\right)^2 & s \in \mathcal{R} \\ t_N &= \begin{cases} 0.5 \cdot \left(\frac{t}{\xi_0}\right)^2 & 0 \leq t < \xi_0 \\ \sqrt{1 - \frac{3\xi_0}{4t}} & \xi_0 \leq t \end{cases} \end{cases}$$

Valet att låta s , d.v.s. x_1 ha tätare diskretisering nära gränsen $|x_1| = \xi_0$ visade



Figur 4.1 Transformationer av normerade storheter då $\xi_0 = 1$. De streckade linjerna är gränslinjer. De punktstreckade är hjälplinjer.

sig senare inte vara motiverat eftersom någon snabb förändring i V m.a.p. s inte uppstod.

De diskreta tillståndsvärden som tabellen skall lagras för har nu valts. Initieringen av tabellen är enkel. Det gäller att $V(N)$ är noll om $\xi(N)$ är mindre än ξ_0 . Eftersom tabellen bara lagras för $\xi < \xi_0$ blir $V(N)$ noll för alla punkter.

Lagring av den optimala styrsignalen Valet av tillstånd att lagra V för har nu diskuterats grundligt. En bieffekt av valet är att den optimala styrsignalen också kommer att erhållas för just dessa tillstånd. Det måste då kontrolleras att styrlagen varierar mest på samma ställe som förlusten och att liknande symmetriegenskaper kan utnyttjas.

Den första frågan kan inte avgöras innan räkningarna utförts. Den andra frågan får ett positivt svar genom följande samband

$$u(-x_1, -x_2) = -u(x_1, x_2)$$

som följer av resonemang liknande de som gav $V(-x_1, -x_2) = V(x_1, x_2)$.

Interpolering i tabeller Eftersom både den minimala delförlusten V och motsvarande styrsignaler u är lagrade i diskreta punkter måste interpolationer och extrapolationer utföras då dessa behövs för andra punkter än de lagrade. Metoden som valts är att till de inre punkterna i tabellen ordna polynom som approximerar förlosten/styrsignalen. Varje polynom har anpassats till nio punkter med hjälp av minstakvadratmetoden. Koefficienterna i polynomen lagras undan i speciella koefficientmatriser.

När ett värde på V eller u skall bestämas utnyttjas närmaste tabellpunkt. Motsvarande polynom används sedan för att räkna ut ett approximativt värde. Eftersom polynomen inte anpassats till varandra kommer det att finnas diskontinuiteter där gränsen mellan olika polynomval sker.

Tack vare symmetri och begränsningar finns tabellerna bara lagrade för vissa områden av värdet. Genom att utnyttja symmetri och begränsningar täcks dessa områden in. Dessa symmetriegenskaper är olika för styrsignalen u och den minimala delförlusten V varför olika rutiner måste implementeras.

Uppdateringen När datatyperna och initieringar av desamma är klara är det dags för uppdateringar. Givet nuvarande förlusttabell skall föregående bestämmas med tillhörande styrsignaler. Resultatet används sedan för att erhålla tabeller för tidigare tidpunkter.

Minimeringen För varje tillstånd i de önskade tabellerna skall den optimala styrsignalen hittas och lagras tillsammans med motsvarande minimala förlust. Hur optimeringen skall ske beskrivs av ekvation 4.3. Praktiskt kan optimeringen utföras genom att utföra väntevärdesintegrationen (4.4) för ett antal styrsignaler och välja den styrsignal som ger lägst förlust.

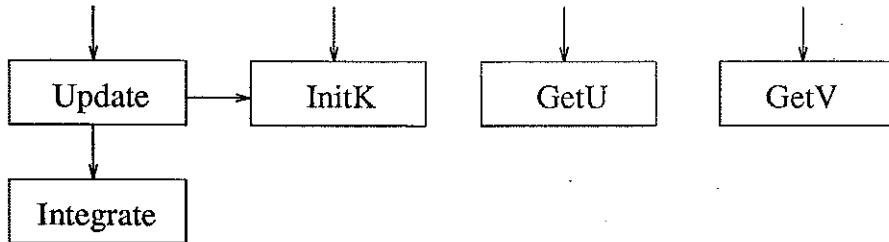
För att uppnå ett bra värde på u har en kvadratisk funktion anpassats till tre (u, V) -punkter för att ge ett bättre värde på styrsignalen. För detta värde har integrationen upprepats för att få ett förbättrat värde på förlosten.

Väntevärdesbestämningen Väntevärdesberäkningen som skall utföras vid optimeringen ges av ekvation 4.4 och består av en enkelintegral m.a.p. bruset $w(k)$. Tätheten för bruset är enligt förutsättning normalfördelat. Detta gör att tätheten är enkel. Argumentet vid integrationen är lika enkel att få fram. Det som behövs är nästa tillstånd. Detta ges direkt av systemet, styrsignalen u och det tillstånd för vilket vi räknar.

Som integrationsmetod har valts Simpsons formel med Richardson extrapolation, se Press (1987).

Programstruktur

Nedan kommer implementeringen av den numeriska optimeringen att diskuteras. Programstrukturen som valts åskådliggörs av figur 4.2. Den administrativa delen har utelämnats och disskuterats inte vidare. Rutinen *Update* är huvudrutan och ger ifrån sig optimala styrsignalen och tillhörande förlustfunktion i tabellform. Genom att anropa rutinen med en förlusttabell kan man upprepa



Figur 4.2 Programstruktur för optimeringsrutinerna

förfarandet och erhålla tabeller för alla tidpunkter. *InitK* skapar en tabell med polynomkoefficienter för approximation i tabellen med förluster och även i tabellen med styrsignaler. *Integrate* utför integrationen.

Vid presentationen av tabellerna vill man att värdena i tabellerna för styrsignalen och förlusten kan läsas. Detta görs av funktionerna *GetU* resp. *GetV* som båda har koefficientmatriser som inparametrar. De olika rutinerna finns listade i appendix.

Resultat av optimeringen

Nu skall vi studera resultaten av den numeriska optimeringen i några fall. För ändamålet har följande tre parameteruppsättningar valts

$$\left\{ \begin{array}{lll} \sigma_w = 0.2 & c = 0.7 & \xi_0 = 1 \\ \sigma_w = 0.5 & c = 0.7 & \xi_0 = 1 \\ \sigma_w = 0.5 & c = 0.95 & \xi_0 = 1 \end{array} \right.$$

Valet av σ_w har i första fallet gjorts så att standardavvikelsen hos utsignalen, då systemet styrs med minimalvariansregulatorn, blir $\xi_0/3$. De andra har valts för att införa mer bruskorrelation och/eller brus.

Dynamisk optimering resulterar i styrlagar och förlustfunktioner för tidpunkterna $N, N - 1 \dots N - k \dots$ tills man slutar iterera. Eftersom tidsfördröjningen är två kan styrsignalerna $u(N)$ samt $u(N - 1)$ inte påverka förlusten och de kan därför väljas godtyckligt. Detta innebär att endast styrsignalerna för $k \leq N - 2$ är av intresse.

Optimeringen har skett med upplösningen 40 punkter per enhet normerad storhet. Alla tre parameteruppsättningarna har använts. Resultaten för $k = N - 2$ samt $k = N - 3$ kan ses i figurerna på de kommande sidorna. Det visade sig att rekursionen konvergerade redan vid $k = N - 3$ så att styrlagarna, inte förlusterna, för $k < N - 3$ blev identiska med styrlagarna för $k = N - 3$.

Som jämförelse används minimalvariansregulatorn som oavsett parameteruppsättning är

$$u(k) = -x_1(k) - 2x_2(k)$$

Figurerna visar att den optimala styrlagen är ekvivalent med minimalvariansstyrlagen om tillståndet befinner sig långt från gränsen d.v.s. om $|x_1(k) + x_2(k)| \ll \xi_0$. Om man däremot närmar sig gränsen kommer den optimala regulatorn att avvika från minimalvarianslösningen. Detta sker genom att beloppet på styrsignalen minskar jämfört med minimalvarians.

En bit utanför gränsen är förlusten nästan oberoende av styrsignal eftersom det då är i princip omöjligt att klara sig undan en gränsöverträdelse. Denna okänslighet för styrsignal är orsaken till att numeriska problem uppstår en bit efter gränsen. Optimeringsresultaten för sådana tillstånd har därför trunkerats bort. Inverkan ökar med antalet iterationer. Detta leder till att numeriska problem uppstår tidigare vid $k = N - 3$ än vid $k = N - 2$.

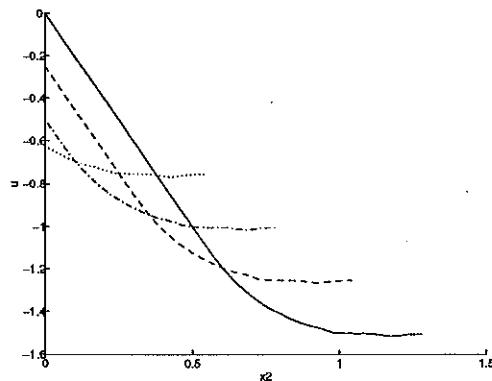
En jämförelse mellan styrstrategierna för $k = N - 2$ och $k = N - 3$ visar att avvikelsen från minimalvarians är större då $k = N - 3$. Inverkan av brusets standardavvikelse σ_w är att punkten där avvikelsen från minimalvariansregulatorn börjar kommer tidigare med större standardavvikelse. Då styrstrategin för $c = 0.95$ jämförs med den för $c = 0.7$ ser vi att punkten där avvikelsen sker inträffar tidigare för $c = 0.95$ samtidigt som krökningen är mindre.

I nästa kapitel kommer den optimala styrstrategin att studeras ytterligare genom simuleringar.

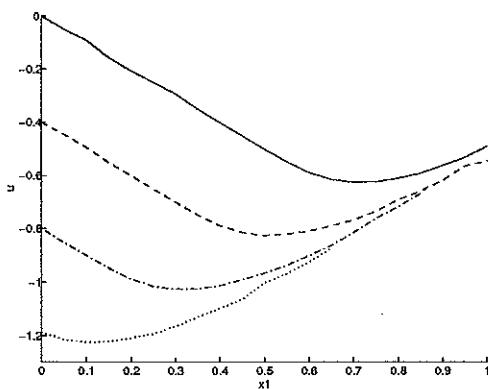
4.5 Sammanfattning

I detta kapitel har fullständig tillståndsinformation studerats vilket leder till att Bellmans ekvation kan förenklas. Detta studerades i ett exempel hämtat från Hansson (1993) som kunde lösas analytiskt då tidsfördröjningen var ett. Den optimala regulatorn blev då identisk med minimalvariansregulatorn. Detta inspirerade oss till att formulera ett exempel med tidsfördröjning två. Som system infördes en dubbelintegrator med färgat brus. Detta exempel verkade omöjligt att lösa analytiskt. Problemet löstes därför numeriskt och resulterade i en optimal regulator som avvek från minimalvariansregulatorn. Förenklat kan man säga att skillnaden mellan den optimala regulatorn och minimalvariansregulatorn låg däri att den förra hade lägre förstärkning då tillståndet var nära den kritiska gränsen.

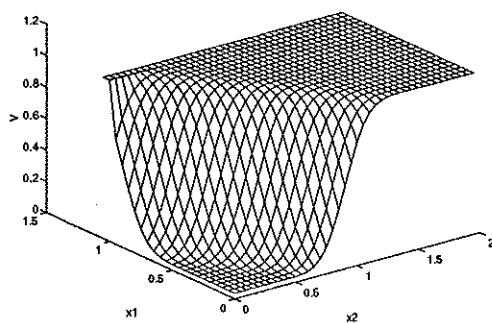
I figurerna nedan gäller $\sigma_w = 0.2$, $c = 0.7$ och $k = N - 2$.



Figur 4.3 Styrignalen u som funktion av x_2 för x_1 -värdena 0 (heldragen), 0.25 (streckad), 0.5 (punktstreckad) och 0.75 (prickad)

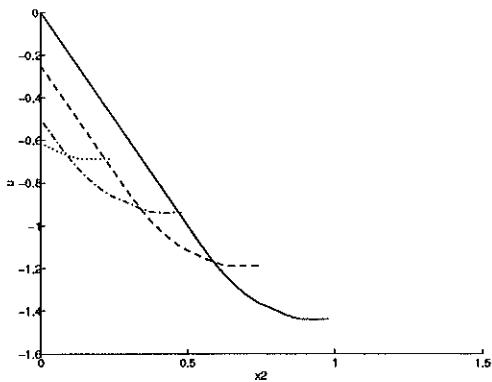


Figur 4.4 Styrignalen u som funktion av x_1 för x_2 -värdena 0 (heldragen), 0.2 (streckad), 0.4 (punktstreckad), 0.6 (prickad)

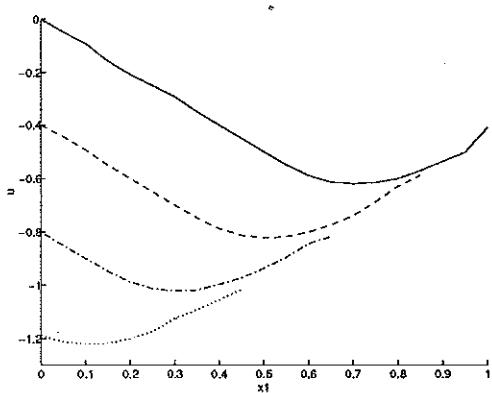


Figur 4.5 Förlustfunktionen V i en kvadrant då $k = N - 2$

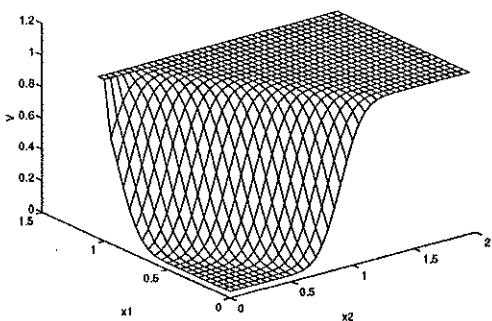
I figurerna nedan gäller $\sigma_w = 0.2$, $c = 0.7$ och $k = N - 3$.



Figur 4.6 Styrsignalen u som funktion av x_2 för x_1 -värdena 0 (heldragen), 0.25 (streckad), 0.5 (punktstreckad) och 0.75 (prickad)

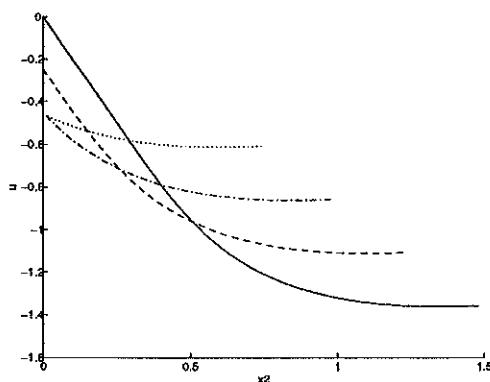


Figur 4.7 Styrsignalen u som funktion av x_1 för x_2 -värdena 0 (heldragen), 0.2 (streckad), 0.4 (punktstreckad), 0.6 (prickad)

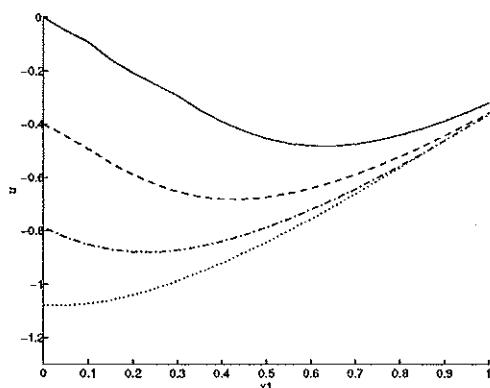


Figur 4.8 Förlustfunktionen V i en kvadrant då $k = N - 3$

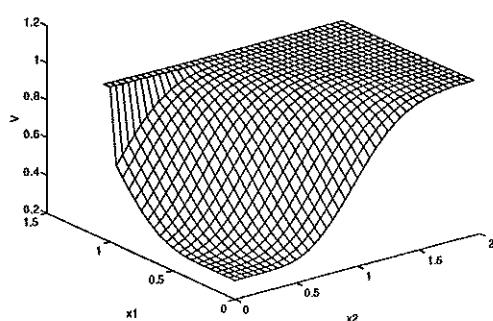
I figurerna nedan gäller $\sigma_w = 0.5$, $c = 0.7$ och $k = N - 2$.



Figur 4.9 Styrsignalen u som funktion av x_2 för x_1 -värdena 0 (heldragen), 0.25 (streckad), 0.5 (punktstreckad) och 0.75 (prickad)

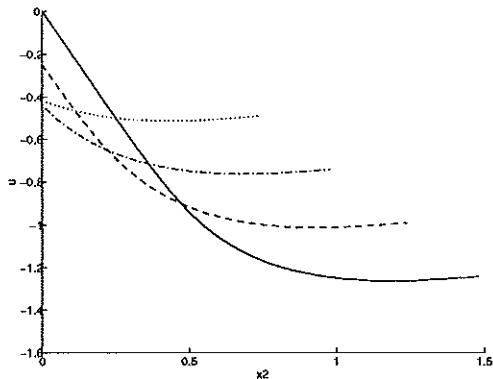


Figur 4.10 Styrsignalen u som funktion av x_1 för x_2 -värdena 0 (heldragen), 0.2 (streckad), 0.4 (punktstreckad), 0.6 (prickad)

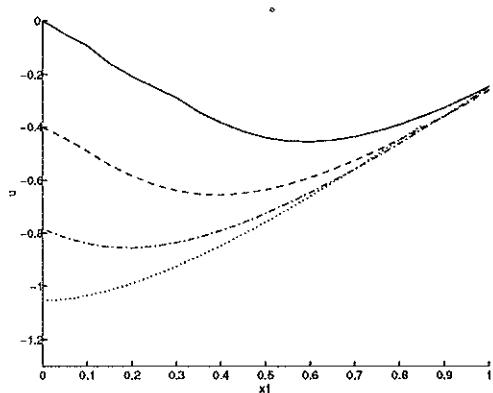


Figur 4.11 Förlustfunktionen V i en kvadrant då $k = N - 2$

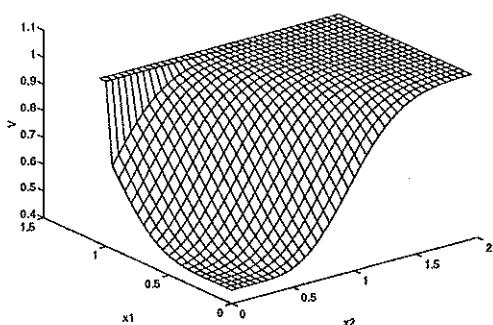
I figurerna nedan gäller $\sigma_w = 0.5$, $c = 0.7$ och $k = N - 3$.



Figur 4.12 Styrsignalen u som funktion av x_2 för x_1 -värdena 0 (heldragen), 0.25 (streckad), 0.5 (punktstreckad) och 0.75 (prickad)

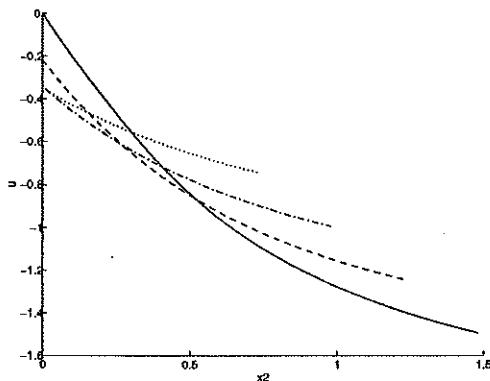


Figur 4.13 Styrsignalen u som funktion av x_1 för x_2 -värdena 0 (heldragen), 0.2 (streckad), 0.4 (punktstreckad), 0.6 (prickad)

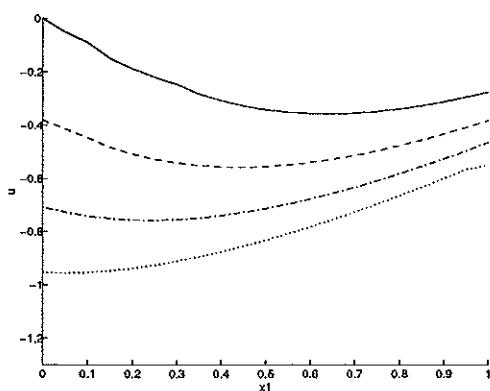


Figur 4.14 Förlustfunktionen V i en kvadrant då $k = N - 3$

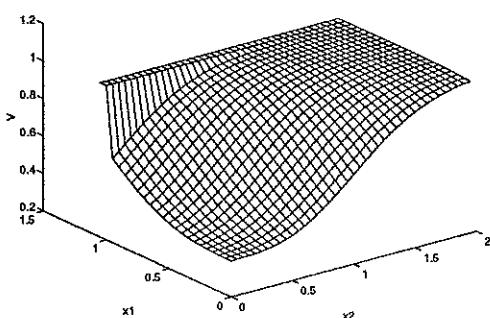
I figurerna nedan gäller $\sigma_w = 0.5$, $c = 0.95$ och $k = N - 2$.



Figur 4.15 Styrsignalen u som funktion av x_2 för x_1 -värdena 0 (heldragen), 0.25 (streckad), 0.5 (punktstreckad) och 0.75 (prickad)

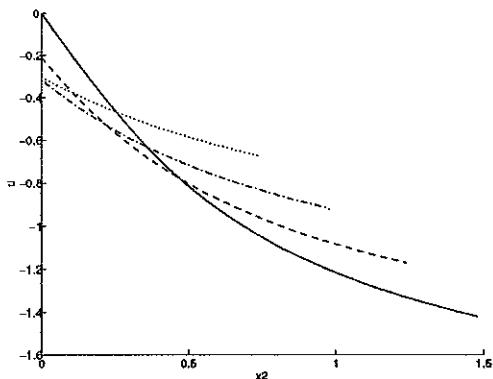


Figur 4.16 Styrsignalen u som funktion av x_1 för x_2 -värdena 0 (heldragen), 0.2 (streckad), 0.4 (punktstreckad), 0.6 (prickad)

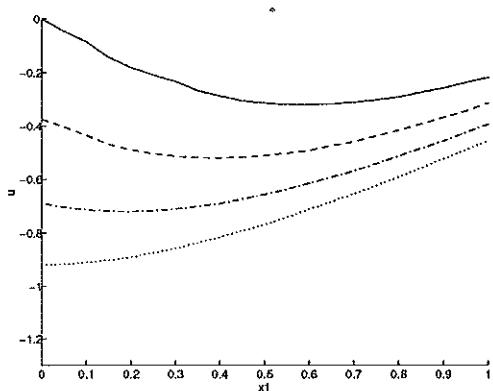


Figur 4.17 Förlustfunktionen V i en kvadrant då $k = N - 2$

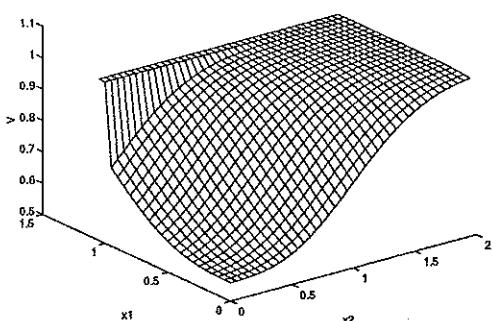
I figurerna nedan gäller $\sigma_w = 0.5$, $c = 0.95$ och $k = N - 3$.



Figur 4.18 Styrsignalen u som funktion av x_2 för x_1 -värdena 0 (heldragen), 0.25 (streckad), 0.5 (punktstreckad) och 0.75 (prickad)



Figur 4.19 Styrsignalen u som funktion av x_1 för x_2 -värdena 0 (heldragen), 0.2 (streckad), 0.4 (punktstreckad), 0.6 (prickad)



Figur 4.20 Förlustfunktionen V i en kvadrant då $k = N - 3$

5. Simuleringsar

I detta kapitel skall den numeriskt erhållna optimala regulatorn i föregående kapitel undersökas genom simuleringar. Vid simuleringarna jämförs den optimala regulatorn med några approximativa regulatorer.

I avsnitt 5.1 sker en kort presentation av de approximativa regulatorerna. Därefter presenteras och diskuteras simuleringarna i avsnitt 5.2. En utförlig presentation av de approximativa regulatorerna ges i de sista avsnitten i kapitlet.

5.1 Approximativa regulatorer

Innan simuleringarna utförs skall en introduktion till de approximativa regulatorerna ges. Dessa regulatorer kommer sedan att användas som jämförelse till den numeriskt beräknade optimala regulatorn vid simuleringarna.

Olika approximationer av förlustfunktionen

De approximativa regulatorerna erhålls genom att minimera en annan förlustfunktion än den som den optimala regulatorn minimerar. De approximativa regulatorernas förlustfunktioner kan tolkas som approximationer till den optimala regulatorns förlustfunktion. Låt oss därför först skriva upp den förlustfunktion som den optimala regulatorn minimerar och studera den.

$$J(u(\cdot)) = P\{\xi(N) \geq \xi_0\}$$

Genom att utnyttja hur maximumet uppdateras kan förlustfunktionen skrivas om som

$$J = P(|x_1(N)| > \xi_0 \vee |x_1(N-1)| > \xi_0 \vee \dots |x_1(0)| > \xi_0)$$

Genom att utnyttja följande samband

$$P(A \vee B) = P(A, \bar{B}) + P(B)$$

upprepade gånger följer att förlusten kan tecknas som

$$J = \sum_{k=0}^N P(|x_1(k)| > \xi_0, |x_1(j)| < \xi_0, 0 \leq j \leq k-1)$$

Denna förlustfunktion skall minimeras m.a.p. styrsignalerna. Detta innebär att vi vill utföra minimeringen

$$\min_{u(i): 0 \leq i \leq N} J$$

Styrsignalen $u(k)$ skall bestämmas baserat på den vid samma tidpunkt tillgängliga informationen $\mathcal{Y}(k)$. Den tillgängliga informationen kan vid fullständig tillståndsinformation representeras av det utökade tillståndet.

Det system vi studerar är en dubbelintegrator vilket innebär att tidsfördröjningen är två sampel. Genom att minimera över två sampel kommer bara

en styrsignal att kunna påverka förlusten. Detta är en första approximation och kan införas i ekvationerna genom att sätta $N = 2$. Då erhålls en summa av tre sannolikheter. De två första kan inte påverkas av styrsignalen. Sammanfattningsvis skall följande minimering utföras

$$\min_{u(0)} P(|x_1(2)| > \xi_0, |x_1(1)| < \xi_0, |x_1(0)| < \xi_0)$$

Tillståndet vid tiden noll beror inte av styrsignalen. Detta gör att vi kan betinga sannolikheten i minimeringsuttrycket m.a.p. tillståndet utan att påverka optimeringsresultatet. Detta är en fördel eftersom styrsignalen skall bestämmas utgående från ett känt tillstånd. Formellt skulle ordet utökat tillstånd ha använts i meningarna ovan med vid tiden noll räcker tillståndet för att veta även det utökade tillståndet eftersom $\xi(0) = |x_1(0)|$. Vi skriver alltså om minimeringen som

$$\min_{u(0)} P(|x_1(2)| > \xi_0, |x_1(1)| < \xi_0, |x_1(0)| < \xi_0, \text{ given } x_1(0), x_2(0))$$

När vi känner tillståndet vid tidpunkten noll kan vi direkt sluta oss till att vi inte kan göra något om vi vet att $|x_1(0)| > \xi_0$. Om så är fallet kan vi välja styrsignal på godtyckligt sätt. Speciellt kan vi välja den så att den minimerar den sannolikhet som minimeras då den är innanför gränsen. Vi kan alltså istället för den ovanstående minimeringen utföra minimeringen

$$\min_{u(0)} P(|x_1(2)| > \xi_0, |x_1(1)| < \xi_0, \text{ given } x_1(0), x_2(0)) \quad (5.5)$$

Att minimera över två sampel är en första approximation om $N > 2$. Denna leder till den olinjära minimaluppkorsningsregulatorn. Genom att postulera en linjär styrlag erhålls den linjära minimaluppkorsningsregulatorn. Genom att postulera en linjär styrlag vid minimeringen

$$\min_{u(0)} P(x_1(2) > \xi_0, x_1(1) < \xi_0, \text{ given } x_1(0), x_2(0))$$

erhålls en ensidig linjär minimaluppkorsningsregulator. Denna kallas i den fortsatta framställningen för linjär minimaluppkorsningsregulator.

Utgående från ekvation (5.5) kan man tänka sig att istället utföra minimeringen

$$\min_{u(0)} P(|x_1(2)| > \xi_0, \text{ given } x_1(0), x_2(0))$$

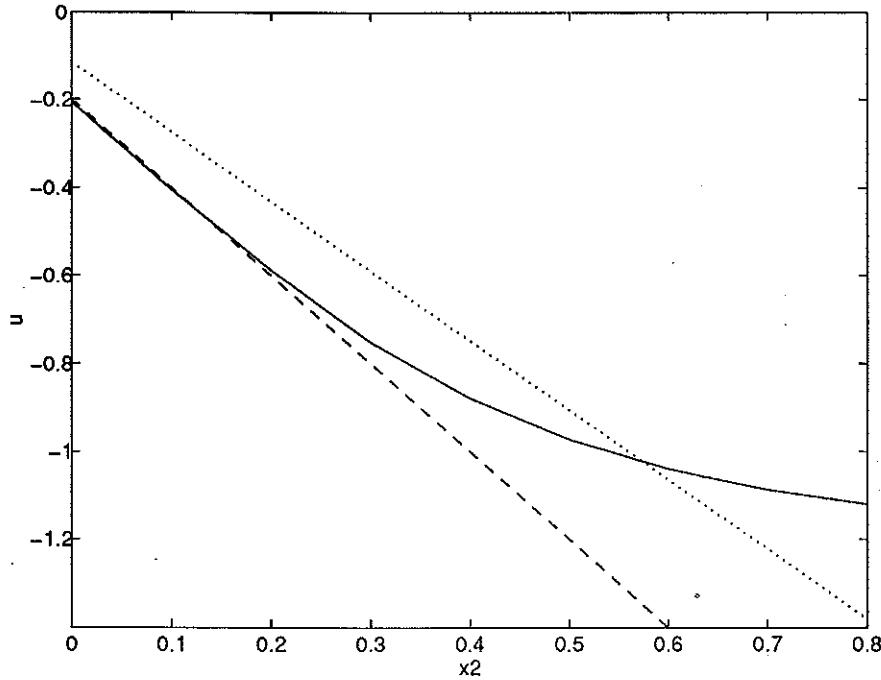
Denna approximation är god då ξ_0 är stor jämfört med standardavvikelse för x_1 . Regulatorn som minimerar detta uttryck är den välkända minimalvariansregulatorn.

Resulterande styrlagar

De approximativa styrlagarna ges i slutet av kapitlet. Den optimala styrlagen gavs i föregående kapitel. Här följer därför bara en kort presentation av styrlagarnas utseende. Figur 5.1 visar styrsignalerna för de olika regulatorerna som funktion av x_2 då tillståndet $x_1 = 0.2$. Den parameteruppsättning som används är

$$\sigma_w = 0.5 \quad N = 2 \quad c = 0.7$$

Den streckade linjen är minimalvariansregulatorn. Punktlinjen är den linjära minimaluppkorsningsregulatorn. Slutligen är den icke linjära styrlagen den optimala och den sammanfaller för detta fall, då $N = 2$, med den olinjära minimaluppkorsningsregulatorn.



Figur 5.1 Styrlagar då $x_1 = 0.2$ för linjär minimaluppkorsning, punktlinjen, optimal regulator, heldragen, minimal varians, streckad.

5.2 Simuleringar

Vid simuleringarna kommer en jämförelse med minimalvarians och den linjära minimaluppkorsningsregulatorn att göras. Den olinjära minimaluppkorsningsregulatorn kommer iste att simuleras eftersom den är nästan identisk med den optimala lösningen. Då $N = 2$ är de identiska. Simuleringarna har utförts i MATLAB med programmet *Simu* som finns i Appendix. Detta program simulerar systemet reglerat med den optimala regulatorn eller en linjär regulator. Olika simuleringar kan ges samma brussekvenser. Detta har använts. Simuleringarna har utförts för två av de i förra kapitlet studerade parameteruppsättningarna. Dessa är

$$\begin{cases} \sigma_w = 0.2 & c = 0.7 & \xi_0 = 1 \\ \sigma_w = 0.5 & c = 0.7 & \xi_0 = 1 \end{cases}$$

Simulering över två sampel

Först skall vi simulera de olika regulatorerna när minimeringshorisonten är lika med systemets tidsfördröjning, d.v.s. $N = 2$. I detta fallet kan bara en styrsignal påverka förlusten. För varje starttillstånd har 20000 simuleringar utförts. Resultaten kan ses i tabellerna på de följande två sidorna.

Den översta tabellen visar värdet på styrignalen från regulatorerna vid ett antal olika tillstånd.

Mittentabellen visar de ur simuleringarna skattade förlusterna samt den beräknade optimala förlusten. Skattningarna av förlusten J baseras på alla 20000 simuleringarna. Dessa skattningar avviker från det exakta värdet av två anledningar. Den första är att de optimala styrlagarna inte beräknats exakt. Den andra är att simuleringarna i sig själv ger viss variation. Det är enkelt att

visa att denna variation har följande standardavvikelse

$$\sigma_J = \sqrt{\frac{J - J^2}{M}}$$

Ekvationen säger att simuleringsfelet ökar med förlusten. Vid $M = 20000$ simuleringar gäller t.e.x. att $J = 0.01$ ger standardavvikelsen 0.0007 och $J = 0.1$ ger 0.002. Vid en jämförelse med den numeriskt beräknade förlusten måste hänsyn också tas till onoggrannhet i J från optimeringsberäkningarna.

Den nedersta tabellen jämför två regulatorer åt gången. Beteckningen $f - (l) - s$ i tabellen innebär att den förstnämnda regulatorn var bättre än den andra f gånger vad gäller förlust. På motsvarande sätt betecknar s antalet gånger den andra regulatorn var bättre än den förstnämnda. Storheten l visar antalet gånger regulatorerna hade samma förlust.

Diskussion över resultatet för de två parameteruppsättningarna finns under respektive tabeller.

Simulering då $\sigma_w = 0.2$

Styrsignaler

x_1	x_2	$u_{optimal}$	$u_{minuppk}$	u_{minvar}
0	0	0.0000	0.000	0
0.1	0.1	-0.2929	-0.216	-0.3
0.2	0.2	-0.6000	-0.432	-0.6
0.3	0.3	-0.8883	-0.648	-0.9
0.35	0.35	-0.9756	-0.756	-1.05
0.4	0.4	-1.0121	-0.864	-1.2
0.45	0.45	-1.0169	-0.972	-1.35
0.5	0.5	-0.9986	-1.080	-1.5
0.3	0.5	-1.1106	-0.964	-1.3
0.5	0.3	-0.9107	-0.764	-1.1

Skattade och beräknade förluster

x_1	x_2	$\hat{J}_{optimal}$	$\hat{J}_{minuppk}$	\hat{J}_{minvar}	$J_{beräknad}$
0	0	0.0060	0.0060	0.0060	0.0066
0.1	0.1	0.0070	0.0069	0.0069	0.0066
0.2	0.2	0.0070	0.0069	0.0070	0.0065
0.3	0.3	0.0075	0.0073	0.0076	0.0073
0.35	0.35	0.0165	0.0173	0.0178	0.0188
0.4	0.4	0.0784	0.0802	0.0807	0.0775
0.45	0.45	0.2387	0.2410	0.2416	0.2377
0.5	0.5	0.5012	0.5044	0.5049	0.5000
0.3	0.5	0.0775	0.0791	0.0796	0.0775
0.5	0.3	0.0755	0.0770	0.0776	0.0775

Jämförelse mellan regulatorerna

x_1	x_2	opt-minuppk	opt-minvar	minuppk-minvar
0	0	0-(20000)-0	0-(20000)-0	0-(20000)-0
0.1	0.1	0-(19999)-1	2-(19993)-5	2-(19994)-4
0.2	0.2	6-(19986)-8	0-(20000)-0	8-(19986)-6
0.3	0.3	1-(19995)-4	9-(19983)-8	13-(19978)-9
0.35	0.35	24-(19968)-8	34-(19958)-8	10-(19990)-0
0.4	0.4	40-(19956)-4	50-(19946)-4	10-(19990)-0
0.45	0.45	45-(19955)-0	57-(19943)-0	12-(19988)-0
0.5	0.5	63-(19936)-1	74-(19925)-1	11-(19989)-0
0.3	0.5	38-(19957)-5	48-(19947)-5	10-(19990)-0
0.5	0.3	34-(19962)-4	47-(19948)-5	13-(19986)-0

Tabellerna ovan visar att förlusten är i det närmaste lika för alla tre regulatorerna vid så här liten varians. Jämförelsetabellen visar dock att den optimala regulatorn är bäst följd av den linjära minimaluppkorsningsregulatorn. Minimalvariansregulatorn är sämst. Effekten ökar ju närmare gränsen tillståndet befinner sig. Notera dock att alla tre i praktiken fungerar nästan lika bra.

Simuleringsresultat då $\sigma_w = 0.5$

Styrsignaler

x_1	x_2	$u_{optimal}$	$u_{minuppk}$	u_{minvar}
0	0	0.0000	0.000	0
0.1	0.1	-0.2930	-0.216	-0.3
0.2	0.2	-0.5875	-0.432	-0.6
0.3	0.3	-0.7790	-0.648	-0.9
0.35	0.35	-0.8225	-0.756	-1.05
0.4	0.4	-0.8393	-0.864	-1.2
0.45	0.45	-0.8368	-0.972	-1.35
0.5	0.5	-0.8199	-1.080	-1.5
0.3	0.5	-0.9391	-0.964	-1.3
0.5	0.3	-0.7390	-0.764	-1.1

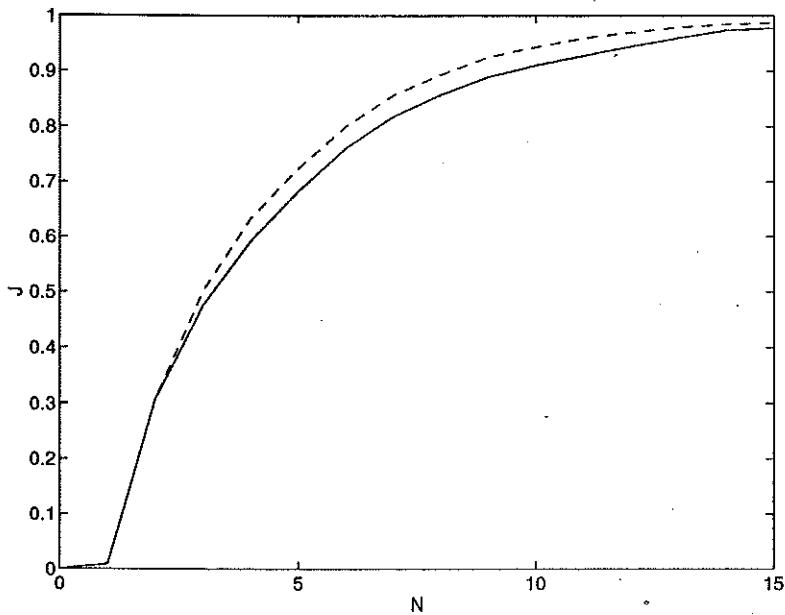
Skattade och beräknade förluster

x_1	x_2	$\hat{J}_{optimal}$	$\hat{J}_{minuppk}$	\hat{J}_{minvar}	$J_{beräknad}$
0	0	0.2780	0.2780	0.2780	0.2767
0.1	0.1	0.2818	0.2816	0.2818	0.2766
0.2	0.2	0.2760	0.2848	0.2761	0.2778
0.3	0.3	0.2992	0.3032	0.3051	0.2989
0.35	0.35	0.3231	0.3265	0.3439	0.3320
0.4	0.4	0.3833	0.3827	0.4242	0.3861
0.45	0.45	0.4587	0.4651	0.5254	0.4605
0.5	0.5	0.5511	0.5658	0.6391	0.5493
0.3	0.5	0.3871	0.3882	0.4274	0.3861
0.5	0.3	0.3871	0.3872	0.4238	0.3861

Jämförelse mellan regulatorerna

x_1	x_2	opt-minuppk	opt-minvar	minuppk-minvar
0	0	0-(20000)-0	0-(20000)-0	0-(20000)-0
0.1	0.1	353-(19289)-358	34-(19932)-34	392-(19221)-387
0.2	0.2	828-(18521)-651	53-(19895)-52	704-(18416)-880
0.3	0.3	551-(18977)-472	528-(19062)-410	1000-(18039)-961
0.35	0.35	275-(19518)-207	975-(18467)-558	1182-(17985)-833
0.4	0.4	65-(19859)-76	1411-(17997)-592	1346-(18138)-516
0.45	0.45	351-(19425)-224	1776-(17782)-442	1425-(18357)-218
0.5	0.5	535-(19224)-241	2058-(17644)-298	1523-(18420)-57
0.3	0.5	85-(19853)-62	1359-(18089)-552	1274-(18236)-490
0.5	0.3	71-(19860)-69	1330-(18074)-596	1259-(18214)-527

Tabellerna visar att nytta med att använda den optimala regulatorn eller den linjära minimaluppkornsningregulatorn har ökat jämfört med förra fallet då variansen var mindre. På samma sätt som tidigare blir skillnaden större ju närmare gränsen tillståndet befinner sig. I detta fallet är det dock en signifikant förbättring nära gränsen. Både den optimala och den linjära minimaluppkornsningregulatorn är väsentligt bättre än minimalvariansregulatorn. För medelstora avstånd är dessa två regulatorer ungefär likvärdiga medan den optimala kan vara något lite bättre riktigt nära gränsen.



Figur 5.2 Förlust som funktion av antal sampel att minimera över. Minimal varians steckad, optimal regulator heldragen.

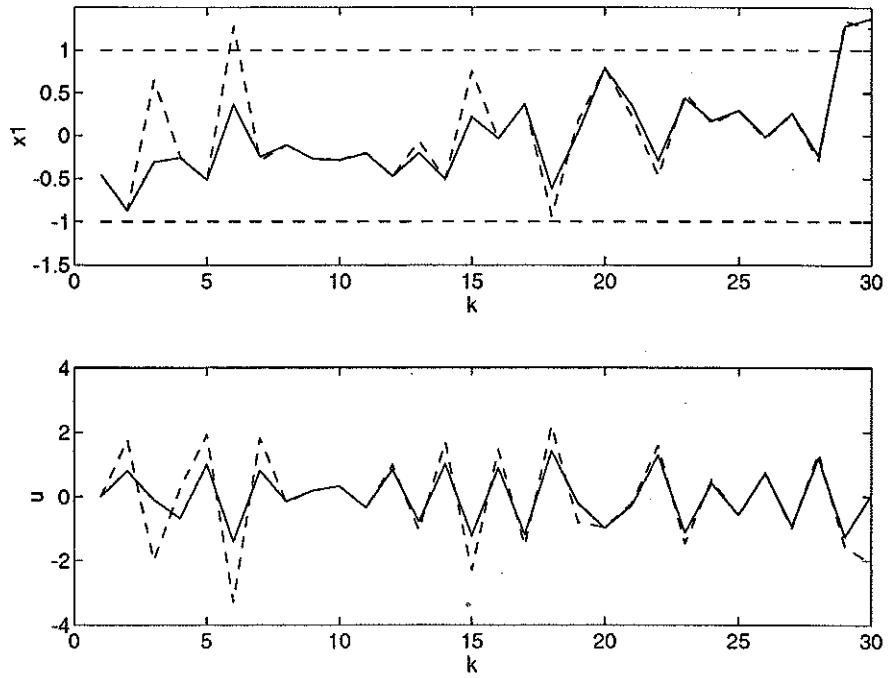
Simulering över många sampel

Hittills har vi studerat hur förlusten beror av tillståndet då minimeringen skett för $N = 2$, d.v.s. då bara en styrsignal har kunnat påverka systemets maximum. För att studera hur förlusten beror av antal sampel att minimera över har 1000 simuleringar utförts. Begynnelsetillståndet var origo och $\sigma_w = 0.5$. Förlusten som funktion av antalet sampel att minimera över kan ses i figur 5.2. Figurens nedre kurva visar förlusten då den optimala styrlagen används. Den övre förlusten då minimalvariansregulatorn används. Tidssimuleringar i två fall kan ses i figur 5.3 och figur 5.4. Figur 5.3 jämför den optimala regulatorn med minimalvariansregulatorn. Man ser att det är mycket variation i x_1 jämfört med det område inom vilket man vill hålla sig. Detta är en förutsättning för att den optimala regulatorn skall vara bättre än minimalvariansregulatorn. Att ha dessa hårdare krav leder till att man ganska snart överskrider gränsen oavsett regulator. Detta kan också ses i figuren. Figur 5.4 jämför den optimala regulatorn med den linjära minimaluppkorsningsregulatorn. Ur båda figurerna kan man notera att både den linjära minimaluppkorsningsregulatorn och den optimala regulatorn kräver mindre styrsignaler än minimalvariansregulatorn.

Simuleringar utfördes även för fallet $\sigma_w = 0.2$ men då blev det ingen synbar skillnad mellan minimalvariansstyrlagen och den optimala styrlagen vad gäller förlustfunktion. En tidssimulering då minimalvariansregulatorn var sämre än den optimala regulatorn kan ses i figur 5.5. Det skall noteras att det är ytterst sällan en sådan effekt går att hitta. Oftast är det t.o.m. så att styrsignalerna är identiska.

5.3 Minimalvariansregulator

Minimalvariansregulatorn har, som namnet antyder, den styrsignal som minimerar variansen hos en signal, i detta fallet $z = x_1$. Detta kan jämföras med den optimala regulatorn vars uppgift är att minimera maximala absolutbelopet hos x_1 under en viss tid.

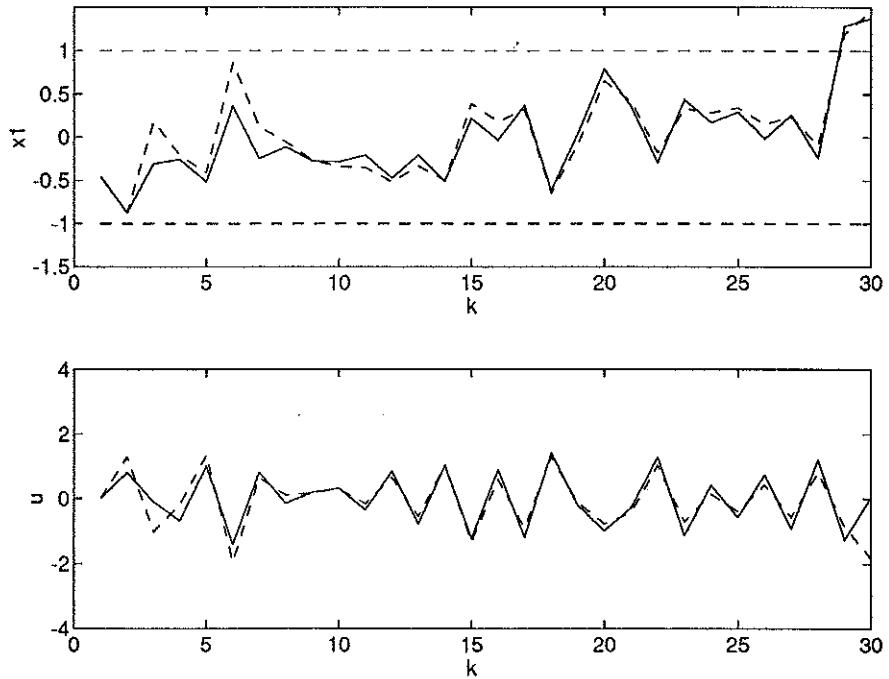


Figur 5.3 Simulerings exempel. Optimal regulator heldragen, minimal varians streckad för $\sigma_w = 0.5$

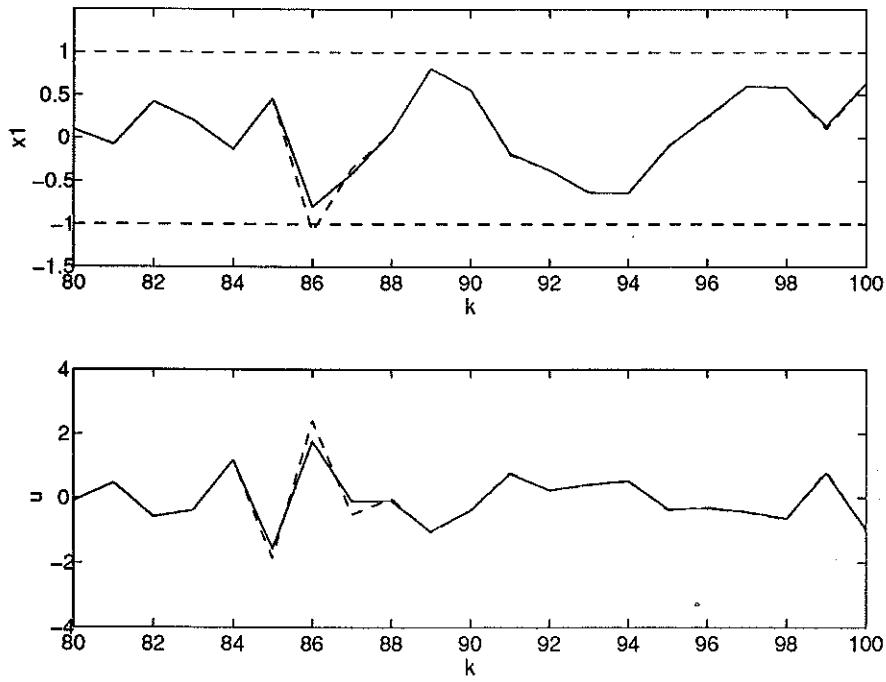
Minimalvariansregulatorns utsignal x_1 kan inte påverkas av styrsignalen förrän efter två sampel. Vi tecknar därför

$$x_1(k+2) = x_1(k) + 2x_2(k) + u(k) + (1+c)w(k) + cw(k+1)$$

När $u(k)$ skall bestämmas finns tillstånden vid samma tidpunkt givna. Ur dessa kan ingen extra information om brusterna bestämmas varför minimal



Figur 5.4 Simuleringsexempel. Optimal regulator heldragen, linjär minimal uppkorsningsregulator streckad, för $\sigma_w = 0.5$



Figur 5.5 Ett unikt fall där optimal regulator klarade sig bättre än minimal varians med $\sigma_w = 0.2$. Minimal varians steckad, optimal regulator heldragen.

varians fås om

$$u(k) = -x_1(k) - 2x_2(k)$$

Detta motsvaras av följande tillståndsåterkopplingsmatris

$$L = [1 \ 2]$$

Denna styrlag ger utsignalens variansen

$$E x_1^2 = \{(1+c)^2 + c^2\} \sigma_w^2 = \{1 + 2c + 2c^2\} \sigma_w^2$$

5.4 Linjär minimaluppkorsningsregulator

I detta avsnitt ges en kort presentation av den linjära minimaluppkorsningsregulatorn. Avsnittet bygger på Hansson (1993).

Problemformulering

Systemet beskrivs av

$$\begin{cases} x(k+1) &= Ax(k) + B_1 u(k) + B_2 v(k) \\ y(k) &= C_1 x(k) + D_1 e(k) \\ z(k) &= C_2 x(k) + D_2 w(k) \end{cases}$$

Bruset är normalfördelat vitt brus med följande kovariansmatris

$$E \left\{ \begin{pmatrix} v \\ e \\ w \end{pmatrix} (v^T \ e^T \ w^T) \right\} = \begin{pmatrix} R_1 & R_{12} & R_{13} \\ R_{12}^T & R_2 & R_{23} \\ R_{13}^T & R_{23}^T & R_3 \end{pmatrix}$$

Problemet kan specificeras som att minimera följande sannolikhet

$$\mu(z_0) = P\{z(0) \leq z_0 \cap z(1) > z_0\}$$

kallad uppkorsningssannolikheten. Minimeringen skall ske m.a.p. en linjär styrslag. Detta kan ske antingen genom att försöka hålla z nedanför gränsen z_0 eller genom att hålla z ovanför gränsen. Eftersom en stabil regulator önskas bör man välja att låta minimeringen ske under restriktionen att utsignalens standardavvikelse σ_z uppfyller villkoret $\sigma_z \leq z_0$. Detta är möjligt om systemet är utsatt för lite brus. Detta antyder att det inte alltid finns en lösning till det specificerade problemet.

Regulatorordesign

Genom att införa

$$\begin{cases} \alpha(k) &= z(k+1) + z(k) \\ \beta(k) &= z(k+1) - z(k) \end{cases}$$

uppstår två oberoende storheter som gör att problemets lösningar kan ringas in. Det kan visas att den bästa linjära regulatorn för problemet finns bland de som minimerar följande förlustfunktion

$$J = E\{(1-\rho)\alpha^2 + \rho\beta^2\} \quad \rho \in [0, 1]$$

Notera att $\rho = 0.5$ motsvarar minimalvariansregulatorn. Förlustfunktion kan skrivas om till ett LQG problem. Först skrivs förlustfunktionen om till

$$J = E\{v^T B_2^T C_2^T C_2 B_2 v\} + \bar{J}$$

där

$$\bar{J} = E\{x^T Q_1 x + 2x^T Q_{12} u + u^T Q_2 u\}$$

Den första termen beror direkt av bruset v och kan därför inte påverkas. Problemet blir därför ekvivalent med att minimera \bar{J} . Detta är ett LQG problem. Matriserna i problemet ges av

$$\begin{aligned} Q_1 &= C_2^T C_2 + A^T C_2^T C_2 A + (1-2\rho)(C_2^T C_2 A + A^T C_2^T C_2) \\ Q_{12} &= (A^T + (1-2\rho)I)C_2^T C_2 B_1 \\ Q_2 &= B_1^T C_2^T C_2 B_1 \end{aligned}$$

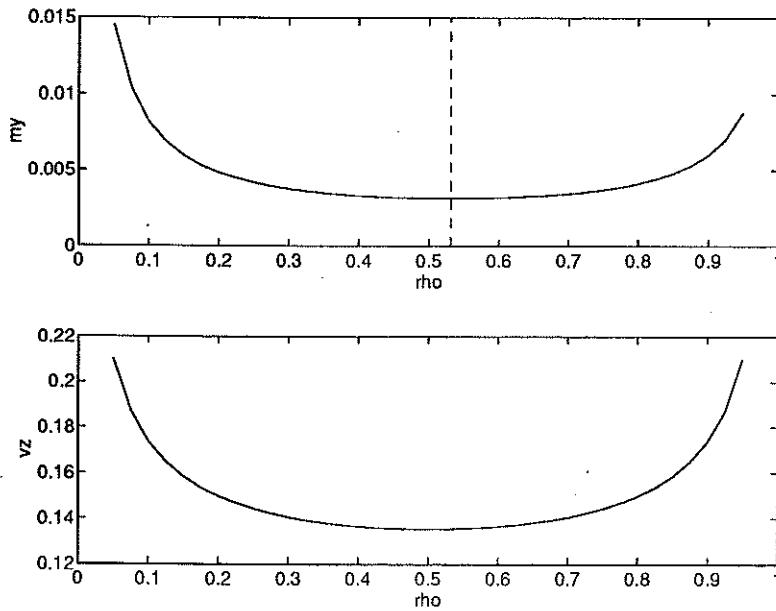
Detta problem skall lösas för i princip alla ρ . Lösningarna kan sedan användas för att beräkna tillhörande uppkorsningssannolikhet enligt

$$\mu = \int_0^\infty \phi(y) \int_{x_l}^{x_u} \phi(x) dx dy$$

där $\phi(x) = 1/\sqrt{2\pi} \cdot \exp(-x^2/2)$ samt gränserna $x_l = (2z_0 - \sigma_\beta y)/\sigma_\alpha$ och $x_u = (2z_0 + \sigma_\beta y)/\sigma_\alpha$. Genom att välja det ρ som minimerar μ är problemet löst. Vid valet måste hänsys tas till bivillkoret $\sigma_z \leq z_0$.

Problemspecifik lösning

Minimaluppkorsningsregulatorn kan användas som en approximation till den optimala regulatorn i vårt ursprungliga problem. Matriserna i minimaluppkorsningsmodellen kan lätt hittas i den tidigare problemspecifikationen.



Figur 5.6 μ och σ_z^2 som funktion av ρ , då $\sigma_w = 0.2$. Streckad linje visar det bästa valet av ρ

Problemet har lösats numeriskt m.h.a. MATLAB. Programmen som används är *intensityd*, *reguld* och *LQregul*, som alla finns i appendix.

Numerisk lösning har skett för två parameteruppsättningar (parametrar i ursprunglig modell)

$$\begin{cases} \sigma_w = 0.2 & c = 0.7 & \xi_0 = 1 \\ \sigma_w = 0.5 & c = 0.7 & \xi_0 = 1 \end{cases}$$

Uppkorsningssannolikheten μ och utsignalsvariansen σ_z^2 för de båda fallen som funktion av ρ kan ses i figur 5.6 och figur 5.7. Det bästa ρ som kan väljas finns markerat med ett lodrätt streck där μ visas som funktion av ρ . I figur 5.7 har gränsen som ges av bivillkoret $\sigma_z \leq z_0$ lagts in där σ_z^2 visas som funktion av ρ . Denna gräns bestämmer lösningen då $\sigma_w = 0.5$. De bästa valen av ρ och tillståndsåterkoppling L är

$$\begin{cases} \rho = 0.531 & L = [0.97 \ 1.97] & \sigma_w = 0.2 \\ \rho = 0.857 & L = [0.58 \ 1.58] & \sigma_w = 0.5 \end{cases}$$

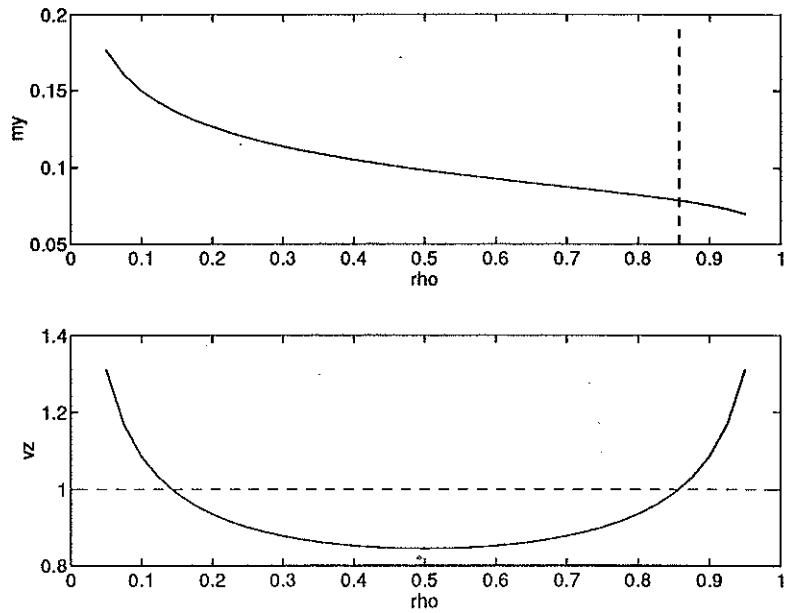
5.5 Olinjär minimaluppkorsningsregulator

I detta avsnitt skall vi först studera exemplet i avsnitt 4.3 utan att välja system. Genom att studera förlustfunktionen kommer en del kunskaper om problemet att erhållas. Dessa kunskaper används sedan för att ta fram den olinjära minimaluppkorsningsregulatoren för systemet som studeras i detta kapitel. Denna regulator visar sig vara den optimala då minimeringen sker över två sampel.

Modell och styrkriterium

Nedan följer en repetition av styrkriterium och valet av maximum. Maximumet uppdateras enligt

$$\xi(k+1) = \max[\xi(k), |z(k+1)|]$$



Figur 5.7 μ och σ_w^2 som funktion av ρ då $\sigma_w = 0.5$. Lodräta streckad linje visar det bästa valet av ρ . Den andra streckade linjen visar maximalt tillåten utsignalvarians.

Detta gör att funktionen g i den generella beskrivningen blir

$$g(z) = |z|$$

Styrkriteriet specificeras av följande förlustfunktion

$$J(u(\cdot)) = P\{\xi(N) \geq \xi_0\}$$

Detta motsvaras i den generella beskrivningen av att man väljer

$$h(k, \xi(k)) = \begin{cases} 0 & , 0 \leq k \leq N-1 \\ \theta(\xi(N) - \xi_0) & , k = N \end{cases}$$

där θ är stegfunktionen definierad av

$$\theta(z) = \begin{cases} 0 & z \leq 0 \\ 1 & z > 0 \end{cases}$$

Styrsignalen skall baseras på det vid samma tidpunkt uppmäta utökade tillståndet $\bar{x} = [x^T \ \xi]^T$.

Analys av kriterium

Styrkriteriet ovan skall nu skrivas om så att insikt i optimeringen kan erhållas. Följande förlustfunktion skall minimeras m.a.p. styrsignalerna

$$J(u(\cdot)) = P\{\xi(N) \geq \xi_0\}$$

Genom att utnyttja hur maximumet uppdateras kan förlustfunktionen skrivas om som

$$J = P(|z(N)| > \xi_0 \vee |z(N-1)| > \xi_0 \vee \dots \vee |z(0)| > \xi_0)$$

Genom att utnyttja följande samband

$$P(A \vee B) = P(A, \bar{B}) + P(B)$$

upprepade gånger följer att förlusten kan tecknas som

$$J = \sum_{k=0}^N P(|z(k)| > \xi_0, |z(j)| < \xi_0, 0 \leq j \leq k-1)$$

Fördröjningens inverkan För att tydliggöra uttrycken inför

$$\gamma(k) = P(|z(k)| > \xi_0, |z(j)| < \xi_0, 0 \leq j \leq k-1)$$

Sannolikheten $\gamma(k)$ beror av de styrsignaler som kan påverka $z(k), z(k-1), \dots$. Låt τ beteckna fördröjningen från styrsignal till z och därmed också till ξ . Denna definition gör att sannolikheten $\gamma(k)$ påverkas av $u(k-\tau)$ och äldre styrsignaler. Uttryckt i $\gamma(k)$ kan förlustfunktionen skrivas som

$$J = \sum_{k=0}^N \gamma(k)$$

Denna skall minimeras m.a.p. styrsignalerna $u(0)$ upp till $u(N)$. P.g.a. fördräjningen kommer de första sannolikheterna inte att kunna påverkas. Av samma anledning kommer de sista styrsignalerna inte att påverka någon sannolikhet. Minimeringen kan därför skrivas som

$$\min_{u(i): 0 \leq i \leq N} J = J_0 + \min_{u(i): 0 \leq i \leq N-\tau} \tilde{J}$$

där vi har

$$J_0 = \sum_{k=0}^{\tau-1} \gamma(k)$$

samt

$$\tilde{J} = \sum_{k=\tau}^N \gamma(k)$$

Styrsignalerna som minimerar \tilde{J} minimerar även J och därför studerar vi bara denna minimering häданefter. D.v.s. vårt problem blir att bestämma u enligt

$$\min_{u(i): 0 \leq i \leq N-\tau} \tilde{J} = \min_{u(i): 0 \leq i \leq N-\tau} \sum_{k=\tau}^N \gamma(k)$$

Införande av betingad sannolikhet När minimeringen skall utföras underlättar det om tillståndet är givet eftersom styrignalen skall bero av detta. Av detta skäl inför den m.a.p. det utökade tillståndet betingade sannolikheten

$$\gamma(k|\bar{x}) = P(|z(k)| > \xi_0, |z(j)| < \xi_0, 0 \leq j \leq k-1 \text{ givet } \bar{x}(k-\tau) = \bar{x})$$

där

$$\bar{x} = [x^T \ \xi]^T$$

betecknar det utökade tillståndet. Med hjälp av γ och väntevärdesbildning över det utökade tillståndet kan vi skriva

$$\gamma(k) = E\{\gamma(k|x, \xi)\}$$

Låt p beteckna en allmän täthetsfunktion. Genom att uttrycka väntevärdet med integration får

$$\gamma(k) = \int p_{\bar{x}(k-\tau)}(x, \xi) \gamma(k|x, \xi) dx d\xi$$

Detta uttryck visar att sannolikheten $\gamma(k)$ beror dels av täheten för det utökade tillståndet och dels av sannolikheten γ givet det utökade tillståndet. I de flesta fallen beror täheten för det utökade tillståndet av alla gamla styrsignaler. Den betingade sannolikheten beror bara av styrsignal och utökat tillstånd vid den givna tidpunkten.

Minimering över en styrsignal Att minimera över en styrsignal motsvarar valet $N = \tau$ d.v.s. att vi minimerar över lika många sampel som tidsfördröjningen är i systemet. Minimeringen i detta fallet förenklas till

$$\min_{u(0)} \tilde{J} = \min_{u(0)} \gamma(\tau)$$

Genom att uttrycka väntevärdet med integration får

$$\min_{u(0)} \int p_{\bar{x}(0)}(x, \xi) \gamma(\tau|x, \xi) dx d\xi$$

Täheten för tillståndet vid tidpunkten noll påverkas ej av styrsignalen vid samma tidpunkt. Härav följer att minimeringen kan flyttas in så att minimeringen blir ekvivalent med att utföra följande minimering

$$\min_{u(0)} \gamma(\tau|x(0), \xi(0))$$

Notera att vi nu skall minimera en storhet som beror av det initiala utökade tillståndet och som därför är stokastisk. Vid tiden noll gäller $\xi(0) = |z(0)|$ vilket innebär att tillståndet räcker för att veta det utökade tillståndet. Vi kan därför skriva om minimeringen som

$$\min_{u(0)} \gamma(\tau|x(0))$$

Explicit kan detta skrivas

$$\min_{u(0)} P(|z(\tau)| > \xi_0, |z(j)| < \xi_0, 0 \leq j \leq \tau - 1 \text{ given } x(0))$$

Om $\xi(0) > \xi_0$ kommer sannolikheten att bli ett oavsett styrsignal. Då $\xi(0) < \xi_0$ skall följande minimering utföras

$$\min_{u(0)} P(|z(\tau)| > \xi_0, |z(j)| < \xi_0, 1 \leq j \leq \tau - 1 \text{ given } x(0))$$

Eftersom styrsignalen i det förra fallet kan väljas godtyckligt kan vi bestämma $u(0)$ från ovanstående uttryck oavsett värdet på $\xi(0)$.

Ovanstående minimering kan alternativt utföras genom att minimera följande uttryck istället

$$\min_{u(0)} P(|z(\tau)| > \xi_0, \text{ given } |z(j)| < \xi_0, 1 \leq j \leq \tau - 1, x(0))$$

Orsaken till detta är att $u(0)$ inte kan påverka något av de äldre tillstånden.

Det implementerade specialfallet

Nu skall vi titta på ett speciellt system nämligen det som diskuterats i kapitlets tidigare avsnitt. Detta ges av

$$\begin{cases} x_1(k+1) = x_1(k) + x_2(k) + c w(k) \\ x_2(k+1) = x_2(k) + w(k) + u(k) \\ z(k) = x_1(k) \end{cases}$$

Notera att $z(k) = x_1(k)$ bestämmer maximumet hos detta system. Notera vidare att systemet som är en dubbelintegrator har tidsfördröjningen $\tau = 2$.

När tidsfördröjningen är två är minimering över två sampel ekvivalent med att ta fram den olinjära minimaluppkorsningsregulatorn. Detta visas och utförs nedan.

Optimeringsuttrycket Minimering över en styrsignal ger för det studerade systemet att vi antingen skall utföra minimeringen

$$\min_{u(k)} P(|x_1(k+2)| > \xi_0, |x_1(k+1)| < \xi_0 \text{ givet } x_1(k), x_2(k)), k \geq 0$$

eller ekvivalent

$$\min_{u(k)} P(|x_1(k+2)| > \xi_0 \text{ givet } |x_1(k+1)| < \xi_0, x_1(k), x_2(k)), k \geq 0$$

med $k = 0$. Den förra minimeringen med godtyckligt k är definitionen på den s.k. minimaluppkorsningsregulatorn. Denna skall nu tas fram genom att använda det andra uttrycket som kan skrivas om som

$$\min_{u(k)} 1 - P(|x_1(k+2)| < \xi_0 \text{ givet } |x_1(k+1)| < \xi_0, x_1(k), x_2(k)), k \geq 0$$

Detta innebär att vi egentligen skall utföra maximeringen

$$\max_{u(k)} P(|x_1(k+2)| < \xi_0 \text{ givet } |x_1(k+1)| < \xi_0, x_1(k), x_2(k))$$

för de tidpunkter då styrignalen kan påverka maximat. Olikheterna kan dessutom uttryckas i de vid samma tidpunkt mätbara tillstånden. För den givna olikheten får vi

$$|x_1(k+1)| < \xi_0 \Leftrightarrow |x_1(k) + x_2(k) + cw(k)| < \xi_0$$

Denna ger villkoret $\varepsilon_1 \leq w(k) \leq \varepsilon_2$. Om $c > 0$ gäller

$$\varepsilon_1 = \frac{-\xi_0 - x_1(k) - x_2(k)}{c} \quad \varepsilon_2 = \frac{\xi_0 - x_1(k) - x_2(k)}{c}$$

Detta villkor uttrycker att tätheten för $w(k)$ baserat på tillgänglig information är trunkerat gaussisk. Av detta kan vi sluta oss till att den givna informationen saknar betydelse om den trunkerade gaussiska fördelningen i stort är identisk med den otrunkrade. För sådana fall bör styrlagen bli ekvivalent med minimalvariansstyrlagen. Trunkeringen saknar betydelse om gränserna ε_1 och ε_2 ligger utanför $-n\sigma_w$ respektive $n\sigma_w$ för ett lämpligt värde på n . Med $n = 2$

kommer t.ex. ungefär fem procent att trunkeras bort. Genom att stoppa in uttrycken för ϵ visar det sig att trunkeringen saknar betydelse om

$$-\xi_0 + n \cdot c \cdot \sigma_w \leq x_1 + x_2 \leq \xi_0 - n \cdot c \cdot \sigma_w$$

Alltså är styrlagen lika med minimal varians om man är nära origo, åtminstone om bruset inte är alltför stort och gränserna inte alltför snäva.

Gränserna inom vilka vi önskar maximal sannolikhet kan också uttryckas i de mätbara tillstånden

$$|x_1(k+2)| < \xi_0 \Leftrightarrow |x_1(k) + 2x_2(k) + u(k) + \zeta| < \xi_0$$

där

$$\zeta = [1+c]w(k) + cw(k+1)$$

införts för att förenkla skrivarbetet. Denna ekvation ger ett villkor $\zeta_1 \leq \zeta \leq \zeta_2$ där

$$\zeta_1 = -\xi_0 - x_1(k) - 2x_2(k) - u(k)$$

och

$$\zeta_2 = \xi_0 - x_1(k) - 2x_2(k) - u(k) \quad (5.6)$$

Optimeringen Nu skall vi hitta den betingade tätheten för brusstorheten ζ eftersom denna tillsammans med styrsignalen indirekt ger oss ett uttryck för sannolikheten som skall minimeras. För ändamålet införs den betingade fördelningsfunktionen

$$F(x) = P(\zeta \leq x | \epsilon_1 < w(k) < \epsilon_2)$$

Denna betingade sannolikhet kan skrivas som

$$F(x) = \frac{P(\zeta \leq x, \epsilon_1 < w(k) < \epsilon_2)}{P(\epsilon_1 < w(k) < \epsilon_2)}$$

Genom att Utnyttja tidigare definitioner erhålls

$$F(x) = \frac{P([1+c]w(k) + cw(k+1) \leq x, \epsilon_1 < w(k) < \epsilon_2)}{P(\epsilon_1 < w(k) < \epsilon_2)}$$

Låt φ beteckna den standardiserade normalfördelningens täthetsfunktion. Eftersom $w(k)$ och $w(k+1)$ är oberoende kan sannolikheten i täljaren skrivas som

$$\int_{\epsilon_1}^{\epsilon_2} \int_{-\infty}^{(x-[1+c]w(k))/c} \varphi\left(\frac{w(k)}{\sigma_w}\right) \varphi\left(\frac{w(k+1)}{\sigma_w}\right) dw(k+1) dw(k)$$

Genom att derivera $F(x)$ m.a.p. x och byta x mot ζ fås

$$p(\zeta | \epsilon_1 < w(k) < \epsilon_2) = \frac{\int_{\epsilon_1}^{\epsilon_2} \varphi\left(\frac{w(k)}{\sigma_w}\right) \frac{1}{c} \varphi\left(\frac{\zeta - [1+c]w(k)}{c \cdot \sigma_w}\right) dw(k)}{\int_{\epsilon_1}^{\epsilon_2} \varphi\left(\frac{w(k)}{\sigma_w}\right) dw(k)}$$

Med dess hjälp kan nu sannolikheten som skall minimeras bestämmas så fort vi bestämt en styrsignal. Man använder därvid att sannolikheten

$$P(|x_1(k+2)| < \xi_0 \text{ given } |x_1(k+1)| < \xi_0, x_1(k), x_2(k))$$

kan skrivas som

$$\int_{\zeta_1}^{\zeta_2} p(\zeta | \varepsilon_1 < w(k) < \varepsilon_2) d\zeta$$

Optimeringen går nu ut på att finna den styrsignalen som maximerar denna sannolikhet. Styrsignalen påverkar gränserna ζ_1 och ζ_2 . Den optimala styrlagen placeras dessa så att tätheten blir så stor som möjligt i intervallet $\zeta_1 \leq \zeta \leq \zeta_2$. Tyvärr är tätheten inte symmetrisk varför maximat inte kan hittas genom att välja väntevärdet på ζ som mittpunkt i integrationsintervallet. Väntevärdet ligger dock nära det optimalet valet av mittpunkt och kan om man så vill användas som en approximation för att enkelt få lite kunskap om styrsignalens utseende. Mittpunkten mellan ζ_1 och ζ_2 kan bestämmas ur ekvation 5.6. Genom att anta att denna mittpunkt skall ligga vid väntevärdet av ζ erhålls följande styrlag

$$u(k) \approx -x_1(k) - 2x_2(k) - \int_{-\infty}^{\infty} \zeta p(\zeta | \varepsilon_1 < w(k) < \varepsilon_2) d\zeta$$

Lösning av specialfallet

För att göra en jämförelse med den numeriskt erhållna optimala regulatorn kommer ovanstående formeler att beräknas numeriskt. Beräkningarna har utnyttjat väntevärdesapproximationen.

Numeriska metoder Integrationerna har utförts numeriskt i en C-rutin. Dessa använder Simpsons metod med Richardson extrapolation, se Press (1987). C-rutinen finns i appendix och heter *Facit*.

För att begränsa det oändliga integrationsintervallet har två egenskaper utnyttjats. Dels att $\varepsilon_1 < w(k) < \varepsilon_2$ och dels att tätheten för $w(k+1)$ utanför $n \cdot \sigma_w$ är försumbar, om n nägerlunda stort. Detta ger gränserna

$$\begin{cases} \zeta_{-\infty} \approx (1+c)\varepsilon_1 - c \cdot n \cdot \sigma_w \\ \zeta_{+\infty} \approx (1+c)\varepsilon_2 + c \cdot n \cdot \sigma_w \end{cases}$$

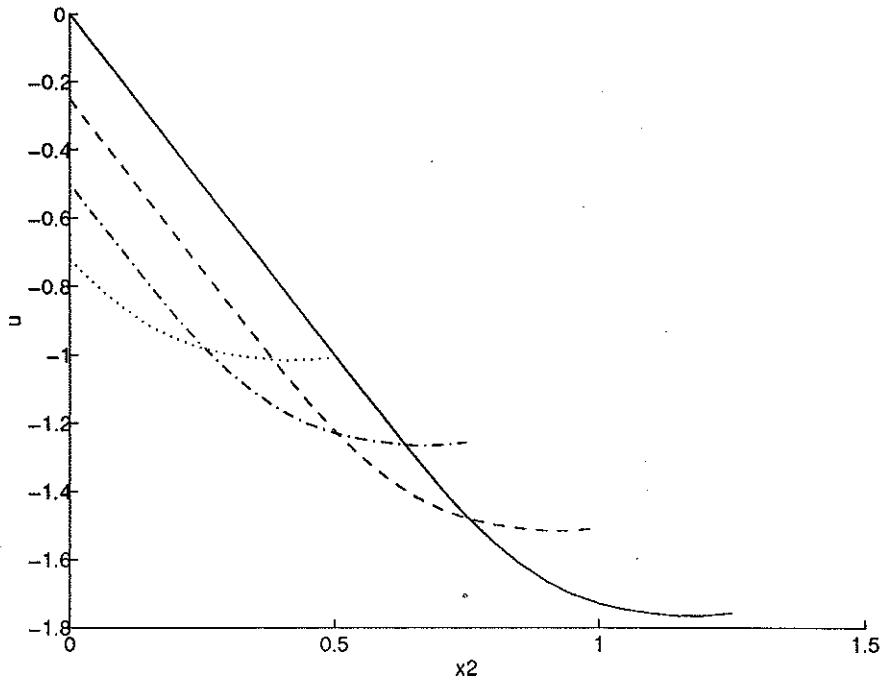
Resultat Beräkningarna gav för fallet $\sigma_w = 0.2$ och $\sigma_u = 0.5$ de styrslagarna som ses i figur 5.8 och figur 5.9 då $c = 0.7$ och $\xi_0 = 1$. Det skall noteras att styrsignalen ovanför $x_1 + x_2 \approx 1 + c \cdot 2\sigma_w$ inte är signifikant på grund av numeriska svårigheter. Den har därför utelämnats.

Figur 5.10 visar en jämförelse mellan den approximativa och den optimala regulatorn. Vi ser att approximationen ger ett systematiskt fel. Något försök att lösa ekvationerna utan väntevärdesapproximationen har inte utförts men bör resultera i den optimala styrlagen.

5.6 Sammanfattning

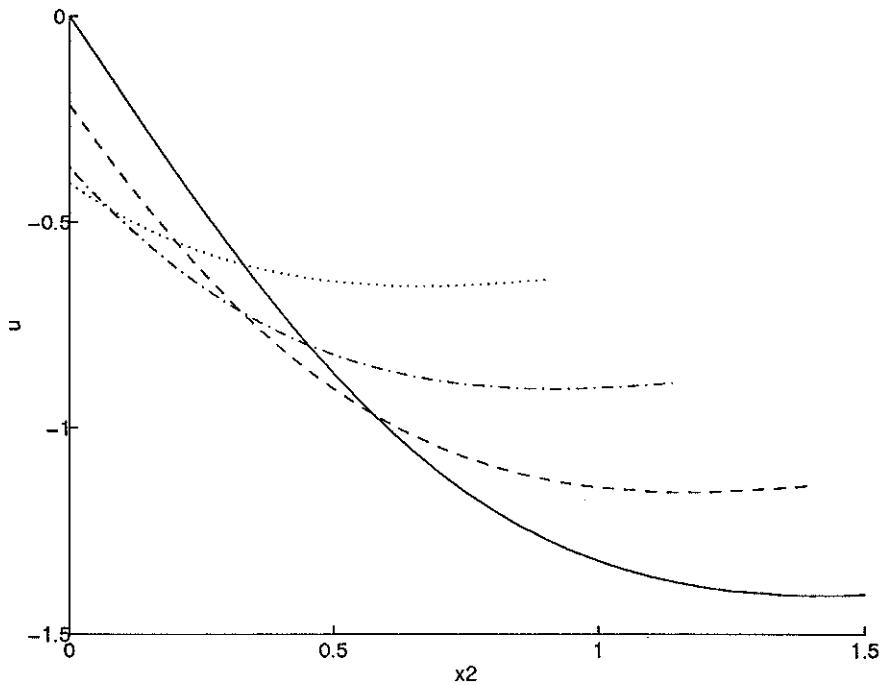
Ett exempel med fullständig tillståndsinformation har studerats. Systemet som studerades var en dubbelintegrator med färgat brus. Problemet gick ut på att få ett maxima att inte komma utanför en given gräns. Den optimala regulatorn till detta problem togs fram i föregående kapitel.

Den optimala regulatorn jämfördes genom simuleringar med ett antal approximativa regulatorer. Dessa simuleringar visade att både den linjära och den olinjära minimaluppkorsningsregulatorn var i stort sett lika bra som den optimala. Jämfört med minimalvariansregulatorn blev det förbättringar om

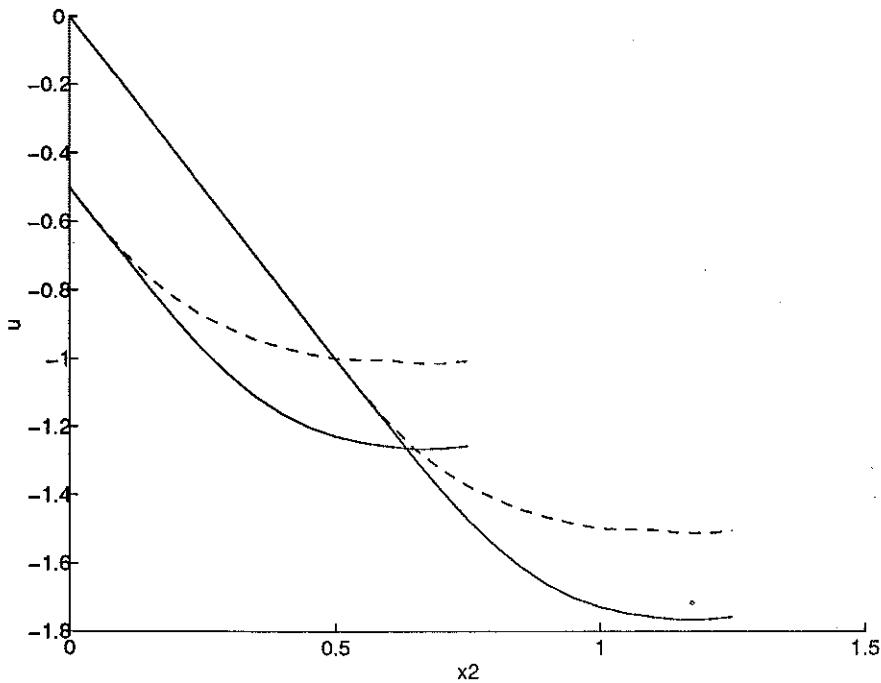


Figur 5.8 Styrsignal som funktion av x_2 för några olika x_1 då $\sigma_w = 0.2$. $x_1 = 0$ för heldragen linje, 0.25 för streckad, 0.5 för punktstreckad och 0.75 för prickad.

tillståndet låg nära gränsen och de ökade ju närmare vi kom gränsen. Förbättringarna var dock mycket små om inte ett rejält brus påverkade systemet. Om så var fallet såg vi i en annan simulering att den optimala regulatorn bara var av intresse om optimeringen skulle ske över ett fåtal sampel. I annat fall skedde en gränsöverträdelse oavsett styrlag. Vidare visade simuleringarna att



Figur 5.9 Styrsignal som funktion av x_2 för några olika x_1 då $\sigma_w = 0.5$. $x_1 = 0$ för heldragen linje, 0.25 för streckad, 0.5 för punktstreckad och 0.75 för prickad.



Figur 5.10 Styrsignal som funktion av x_2 för ett par olika x_1 då $\sigma_w = 0.2$. Approximationen heldragen, optimal regulator streckad

den optimala regulatorn behövde minst styrsignaler medan minimalvarinasregulatorn behövde störst.

Det visade sig att då antalet sampel att minimera över var lika med tidsfördräjningen i systemet kunde man lösa problemet utan att lösa Bellmans ekvation. Lösningen var identisk med den olinjära minimaluppkorsningsregulatorn. Denna metod kräver också numerisk lösning men utan inblandning av förlustfunktionen. Detta minskar komplexiteten väsentligt och bör resultera i att styrsignalerna kan tas fram snabbare. En approximation vid bestämningen av den olinjära minimaluppkorsningsregulatorn reducerar problemet ännu mer så att endast några få integrationer kvarstår. Denna metod ger dock ett systematiskt fel och säger mer om karaktären än om de exakta värdena.

6. Ofullständig tillståndsinformation

Med ofullständig tillståndsinformation menas att tillstånden inte är direkt mätbara utan att bara ett antal mätsignaler korrelerade med tillstånden är mätbara. Med hjälp av dessa kan man filtrera fram betingade fördelningar för tillstånden för att med hjälp av dessa bestämma styrsignalen till systemet.

I avsnitt 6.1 studeras problemställning och filterekvationer. Numerisk lösning genom Bellmans ekvation diskuteras i avsnitt 6.2. I avsnitt 6.3 används simuleringar för att mer ingående studera filtreringsproblemet.

6.1 Det allmänna problemet

Vid ofullständig tillståndsinformation är den givna informationen våra mätningar d.v.s.

$$\mathcal{Y}(k) = \{y(i) : 0 \leq i \leq k\}$$

Bellmans ekvation har följande utseende

$$V(k, p(x(k), \xi(k)|\mathcal{Y}(k))) = \min_{u(k)} E\{h(k, x(k), \xi(k), u(k)) + V(k+1, p(x(k+1), \xi(k+1)|\mathcal{Y}(k+1)))|\mathcal{Y}(k)\}$$

för $0 \leq k \leq N - 1$ med slutvärde

$$V(N, p(x(N), \xi(N)|\mathcal{Y}(N))) = \min_{u(N)} E\{h(N, x(N), \xi(N), u(N))|\mathcal{Y}(N)\}$$

Dessa uttryck beskriver hur man för en given tidpunkt och en given täthet kan bestämma optimal styrsignal och tillhörande förlust. Den givna tätheten ersätter den givna informationen $\mathcal{Y}(k)$. En täthet är enklare att använda än en mängd utsignaler. Väntevärdesbildningen består av två termer. Den första kan lätt uttryckas som en integration

$$\int h(k, x(k), \xi(k), u(k)) p(x(k), \xi(k)|\mathcal{Y}(k)) dx(k) d\xi(k)$$

Den andra termen är något krångligare. Integrandens argument är en täthet som skall baseras på informationen $\mathcal{Y}(k+1)$. Denna täthet går att uttrycka i den givna tätheten, som baseras på $\mathcal{Y}(k)$, och den senaste mätningen $y(k+1)$. Detta visas senare. Mätningen $y(k)$ är den stokastiska storhet över vilken väntevärdet skall räknas

$$\int V(k+1, p(x(k+1), \xi(k+1)|\mathcal{Y}(k+1))) p(y(k+1)|\mathcal{Y}(k)) dy(k+1)$$

Problemet att hitta det utökade tillståndets täthet vid nästa tidpunkt baserat på tätheten nu och den senaste mätningen kallas filtreringsproblemet och beskrivs nedan. Där kommer även tätheten för $y(k+1)|\mathcal{Y}(k)$ att bestämmas.

När väntevärdena kan beräknas gäller det att hitta den styrsignal som minimerar dessa. När detta är klart har man funnit optimal styrsignal med tillhörande förlust.

Filtreringsproblemet

Här skall filtreringsproblemet studeras. Filtrets uppgift är att med hjälp av nya mätsignaler uppdatera tätheten för det utökande tillståndet $\bar{x} = [x^T \ \xi]^T$. Den täthet som skall uppdateras är $p(\bar{x}(k)|\mathcal{Y}(k))$. Den önskade tätheten kan skrivas som

$$p(\bar{x}(k+1)|\mathcal{Y}(k+1)) = \frac{p(y(k+1), \bar{x}(k+1)|\mathcal{Y}(k))}{p(y(k+1)|\mathcal{Y}(k))}$$

När täheterna i högerledet hittats kan man stoppa in $y(k+1)$ och den nya tätheten är funnen. Nämnen i uttrycket behövs inte bara här utan även i väntevärdesberäkningar vid optimeringen. Den kan uttryckas som en integration

$$p(y(k+1)|\mathcal{Y}(k)) = \int p(y(k+1), \bar{x}(k+1)|\mathcal{Y}(k)) d\bar{x}(k+1)$$

Att hitta täheterna har nu reduceras till att hitta den simultana täthetsfunktionen mellan \bar{x} och y . Denna skall vi genast ta fram uttryck för. Eftersom utsignalen bara beror av brus, styrsignal och tillståndet kommer ξ inte att ge någon extra information om utsignalen när man känner tillståndet. Härav följer att

$$p(y(k+1)|\bar{x}(k+1)) = p(y(k+1)|x(k+1))$$

Detta gör att den simultana tätheten kan skrivas som

$$p(y(k+1), \bar{x}(k+1)|\mathcal{Y}(k)) = p(y(k+1)|x(k+1))p(\bar{x}(k+1)|\mathcal{Y}(k))$$

Nu gäller det att hitta de två täheterna i högerledet. Den första ger inga problem eftersom denna ges av Kalmanfiltret. Den andra däremot kan vara svår att uttrycka och får studeras från fall till fall.

För att starta filtreringen behövs $p(\bar{x}(0)|\mathcal{Y}(0))$. Till vårt förfogande har vi $p(\bar{x}(0))$. Med hjälp av ovanstående ekvationer finner vi följande samband

$$p(\bar{x}(0)|\mathcal{Y}(0)) = \frac{p(y(0)|x(0)) \cdot p(\bar{x}(0))}{p(y(0))}$$

Denna tähet måste bestämmas innan filterekvationerna kan användas.

6.2 Numeriska beräkningar

Vi har ovan sett att man kan bestämma den optimala styrlagen genom att stega sig bakåt från slutläget till startläget. Det som krävs är att täheterna är givna så att väntevärdena kan bestämmas.

Vid behandlingen av filtreringsproblemet visade det sig att tätheten vid en viss tidpunkt krävde den initiala fördelningen följt av en mängd filtreringar baserat på alla mätsignaler dittills. För att bestämma en viss tähet krävs en mätsignal samt föregående tähet. Detta innebär en stegning framåt i tiden.

Båda dessa problem kan inte lösas samtidigt. Filtreringen måste veta den givna informationen och kommer typiskt att ske i reell tid. Optimal styrsignal måste därför vara beräknad och lagrad för alla de tätheter som kan uppstå.

Problemet att lagra undan V för alla möjliga täthetsfunktioner är omöjligt och man önskar att på något sätt diskretisera alla möjliga täthetsfunktioner. Detta är bara möjligt om man känner till vilka tätheter som förekommer

och därefter lagrar V för ett urval sådana. Detta är inte enkelt. Kan man där emot parametrisera dessa på något enkelt sätt så blir problemet enklare. Man behöver i så fall bara lagra V för en mängd olika parametervärden istället för för en mängd olika funktioner.

Ett annat alternativ är att lagra V för alla möjliga sekvenser av mätsignaler istället för att lagra för alla möjliga täthetsfunktioner. Om antal diskretiseringssnivåer är n kommer optimeringen att behöva utföras för n^k punkter vid tidpunkten k . Om man kan hitta en parametrisering av täheten med p parametrar som diskretiseras i n nivåer vardera behöver optimeringen ske för n^p punkter för varje tidpunkt. Metoden att lagra V för ett diskret antal mätsignaler är därför bara rimlig om man vill minimera över ett fåtal sampel så att k inte blir stort. Problemet består inte bara av att lagra förlusten för ett fåtal punkter utan också av att beräkna förlusten. Hur svårt detta är i de båda fallen har inte studerats.

Efter de numeriska beräkningarna finns tabeller med V och tillhörande styrsignal tillgänglig, åtminstone för ett visst antal tätheter. Genom att i reell tid utföra filtreringen och hitta de aktuella täheterna är styrsignalen bestämd och problemet löst.

6.3 Simulering av ett filter

För att studera filtrets natur kommer ett exempel att studeras. Exemplet kan definieras av att i den generella modellen låta

$$g(z) = |z|$$

så att maximumtillståndet uppdateras enligt

$$\xi(k+1) = \max(\xi(k), |z(k+1)|)$$

Initialt gäller

$$\xi(0) = |z(0)|$$

För enkelhetens skull antas att tillståndet är en skalär och att $C_2 > 0$. Genom att teckna fördelningsfunktioner och derivera dessa hittar man de intressanta täheterna. Den simultana fördelningen specificeras av två tätheter. Den första täheten $p(x(k+1) = x, \xi(k+1) = \xi | \mathcal{Y}(k))$ kan skrivas som en summa av två termer. Den ena termen är diraclinjerna

$$[\delta(\xi/C_2 + x) \theta(\xi/C_2 - x) + \delta(\xi/C_2 - x) \theta(\xi/C_2 + x)] \cdot$$

$$\frac{1}{C_2} \iint_{-\infty}^{\xi} \frac{1}{\sigma_v} \varphi\left(\frac{x - \Phi x(k) - \Gamma u(k)}{\sigma_v}\right) p(x(k), \xi(k) | \mathcal{Y}(k)) d\xi(k) dx(k)$$

Den andra är

$$\int \frac{1}{\sigma_v} \varphi\left(\frac{x - \Phi x(k) - \Gamma u(k)}{\sigma_v}\right) p(x(k), \xi(k) = \xi | \mathcal{Y}(k)) dx(k) \quad (6.7)$$

i de fall $|x(k+1)| < \xi(k+1)/C_2$ medan den är 0 utanför. Storheten φ har införts som täheten för den standardiserade normalfördelningen. Diraclinjerna och

begränsningen kommer ur det nära samband som finns mellan x och ξ . Den andra intressanta tätheten ges av Kalman filtret och ser ut som följer

$$p(y(k+1) = y|x(k+1)) = \frac{1}{\sigma_e} \varphi \left(\frac{y - C_1 x(k+1)}{\sigma_e} \right)$$

Dessa två tätheter ger täljaren i den önskade tätheten. Nämnden ges av

$$p(y(k+1) = y|\mathcal{Y}(k)) = \frac{1}{\sigma} \varphi \left(\frac{y - C_1(\Phi \hat{x}(k) - \Gamma u(k))}{\sigma} \right)$$

där $\sigma^2 = C_1 R_1 C_1^T + R_2$. Denna behövs även explicit i väntevärdesberäkningarna under optimeringen. Storheten $\hat{x}(k)$ fås från Kalmanfiltret och kan uttryckas som

$$\hat{x}(k) = E(x(k)|\mathcal{Y}(k)) = \int x(k) p(x(k)|\mathcal{Y}(k)) dx(k)$$

Uttryckt i den givna simultana tätheten fås

$$\hat{x}(k) = \int x(k) \int p(x(k), \xi(k)|\mathcal{Y}(k)) d\xi(k) dx(k)$$

Den simultana fördelningen vid nästa tidpunkt kan nu beräknas givet $y(k+1)$. För att kunna använda filtret saknas nu följande täthetsfunktion

$$p(x(0), \xi(0)) = \delta(\xi(0) - |C_2 x(0)|) \varphi \left(\frac{x(0) - m_0}{\sqrt{R_0}} \right) \quad (6.8)$$

som enligt tidigare kan användas för att bestämma starttätheten $p(\bar{x}(0)|\mathcal{Y}(0))$.

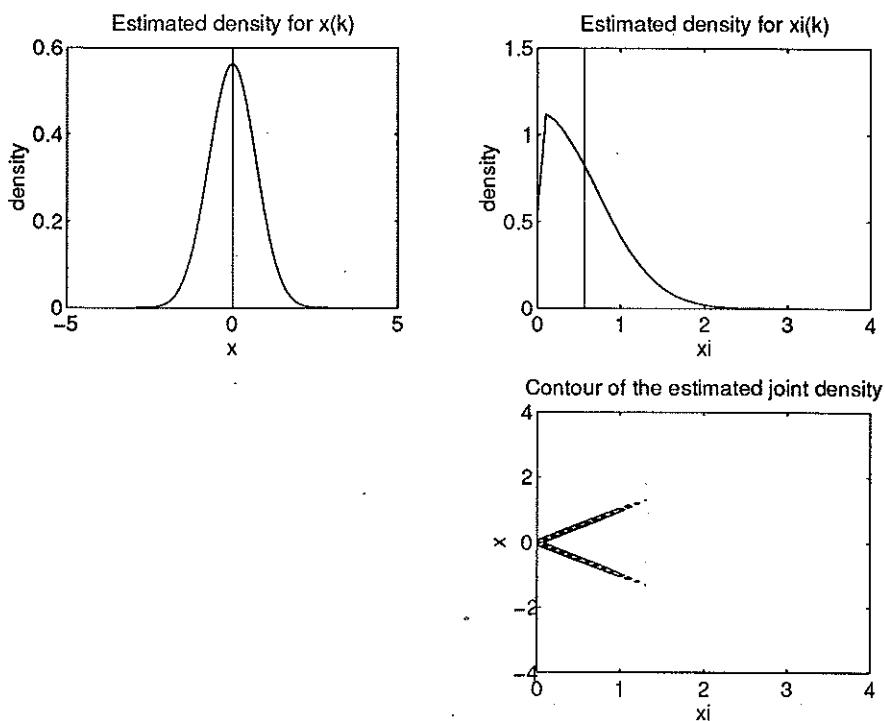
För att få insikt i hur filtret beter sig har simuleringar utförts med ett MATLAB program. Detta program *Maxabsfilter* finns i appendix. I simuleringarna har följande parametrar använts

$$\Phi = 0.5 \quad C_1 = C_2 = 1 \quad R_0 = R_1 = R_2 = 1 \quad m_0 = 0$$

Styrsignalen var hela tiden noll varför värdet på Γ saknar betydelse. Tätheternas centrum flyttar då runt enbart beroende av slumpen. Om en styrsignal skiljd från noll används kommer en överlagrad rörelse orsakad av denna att tillkomma. Tätheternas utseende blir dock likartat.

Täthetsfunktionerna som filtret gav upphov till kan ses i figurerna 6.1 – 6.7 för några olika tidpunkter då tätheterna diskretiseras i axelparallella kvadrater med sidan 0.1. Figur 6.1 visar den tähet som det utökade starttillståndet har då vi inte fått någon mätning. Man kan se att tätheterna består av diraclinjer orsakade av sambandet för initialtillståndet enligt ekvation 6.8. I figur 6.2 kan vi se hur tätheterna vid starttillståndet blir då de baseras på den första mätningen. Den del av den simultana täheten som inte innehåller diracker och som ännu inte har trunkerats d.v.s. uttrycket 6.7 utan trunkering kallas den ”snälla” täheten och kan ses i figurerna.

Figurerna visar att x är normalfordelad precis som väntat eftersom systemet består av Gaussiska brustermer. Dessutom ser vi att ξ inte är Gaussisk vilket också kunde gissats i förväg. Tätheterna för det utökade tillståndet har hela sin korrelation förslagd till gränslinjerna. Innanför gränserna verkar ξ och x sakna korrelation. Detta ses tydligare i figurerna för det obehandlade tähtesbidraget. Dess huvudrikningar är parallella med axlarna vilket innebär att



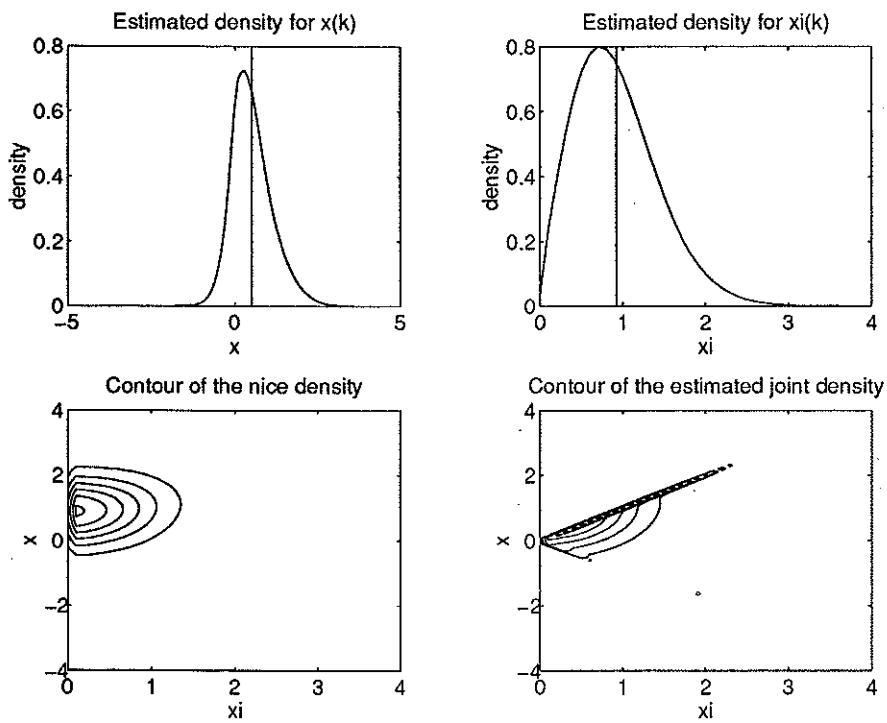
Figur 6.1 De initiala tätigheterna vid tiden noll. De två övre figurerna visar tätigheterna för $x(k)$ och $\xi(k)$. Den nedre visar nivåkurvor till den simultana tätighetsfunktionen mellan $x(k)$ och $\xi(k)$.

t.ex. ett givet ξ inte ändrar förväntat värde på x . På motsvarande sätt verkar det förhålla sig om istället x är givet.

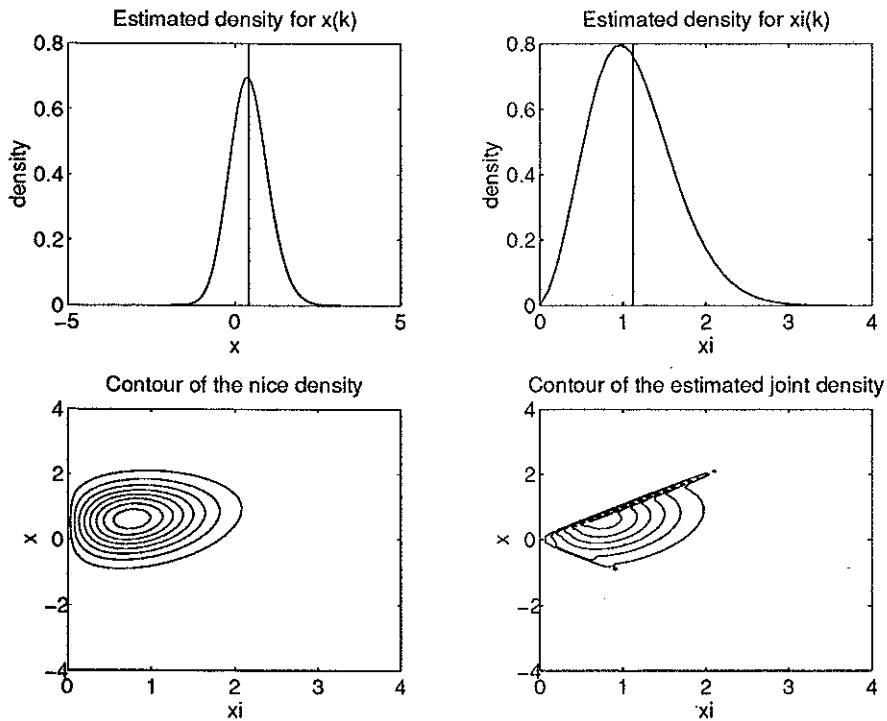
Det bör nämnas att orsaken till att ξ växer så fort är att ingen regulator används. En bra regulator håller kvar ξ vid låga värden vilket gör trunkeringarna och diraclinjerna i den simultana tätheten ännu viktigare. Vid $k = 8$ ser vi att x och ξ börjar bli okorrelerade eftersom diraclinjen börjar försvinna. Detta är intuitivt riktigt eftersom det efter lång tid, speciellt utan regulator inkopplad, borde vara så att x oftast ligger långt från det maximalt uppnådda värdet ξ .

6.4 Sammanfattning

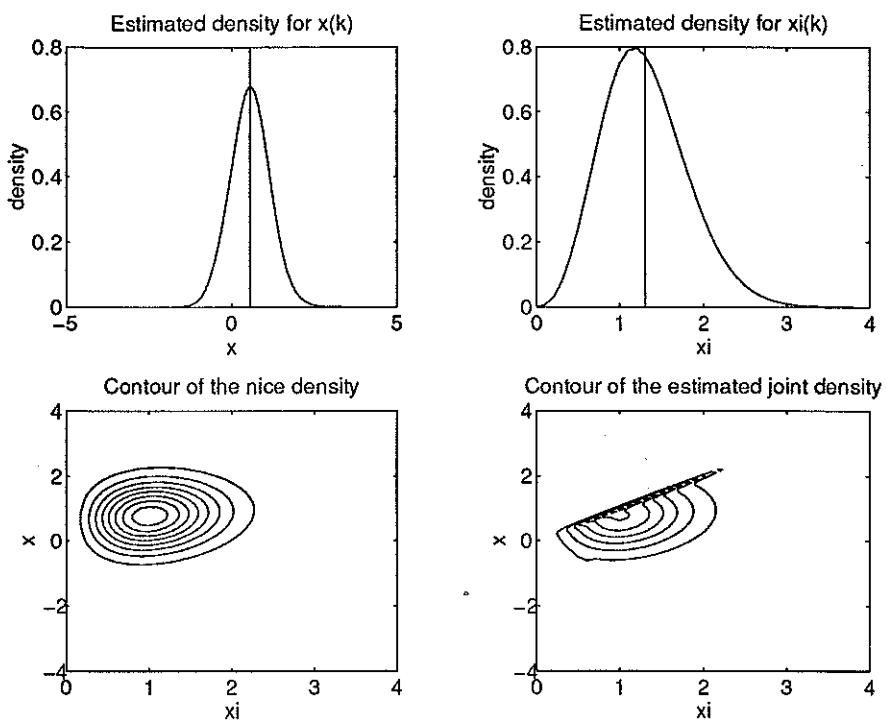
Vid ofullständig tillståndsinformation räkar man ut för att optimeringen som enligt Bellmans ekvationer måste ske bakåt i tiden kräver filtreringar som måste ske framåt i tiden. Detta problem gör att man under optimeringen måste gissa på vilka täteter som filtreringarna kan tänkas ge upphov till. För att få en uppfattning om deras utseende har simuleringar utförts. Dessa ger ideer för hur framtida numeriska beräkningar skall kunna utföras.



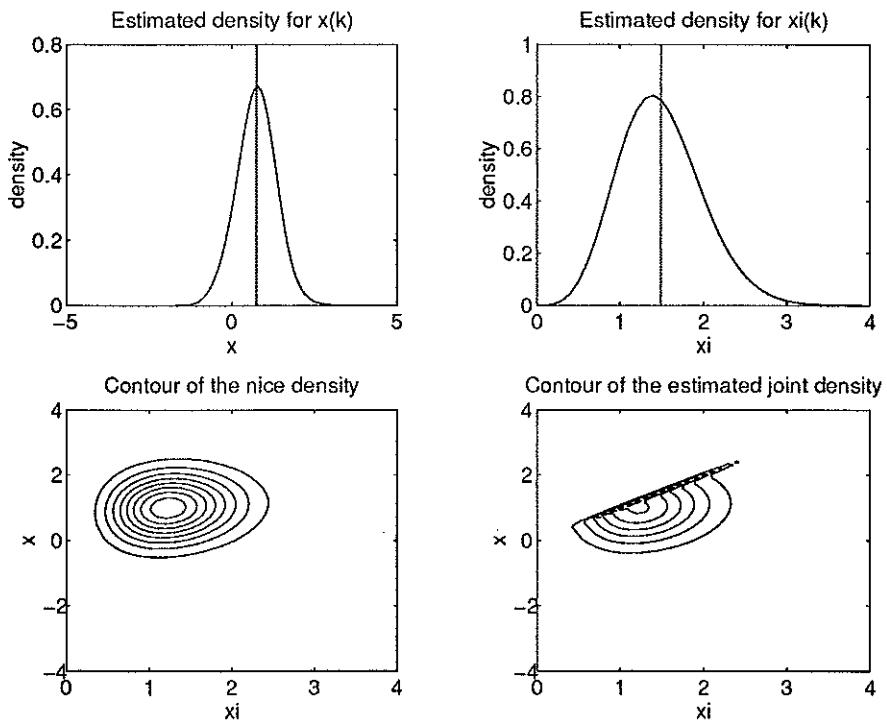
Figur 6.2 Täthetsfunktioner vid tiden 0 baserat på en mätning. De övre figurerna visar tätheterna för $x(k)$ och $\xi(k)$. Den nedre högra visar nivåkurvor till den simultana täthetsfunktionen mellan $x(k)$ och $\xi(k)$. Den nedre vänstra förklaras i texten.



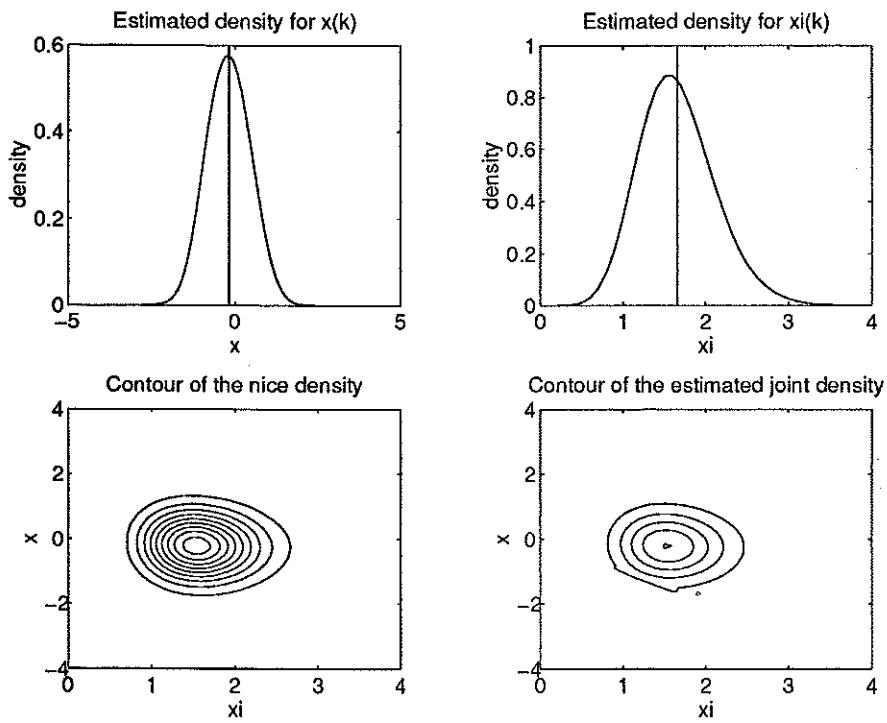
Figur 6.3 Täthetsfunktioner vid tiden 1 baserat på mätningarna. De övre figurerna visar tätheterna för $x(k)$ och $\xi(k)$. Den nedre högra visar nivåkurvor till den simultana täthetsfunktionen mellan $x(k)$ och $\xi(k)$. Den nedre vänstra förklaras i texten.



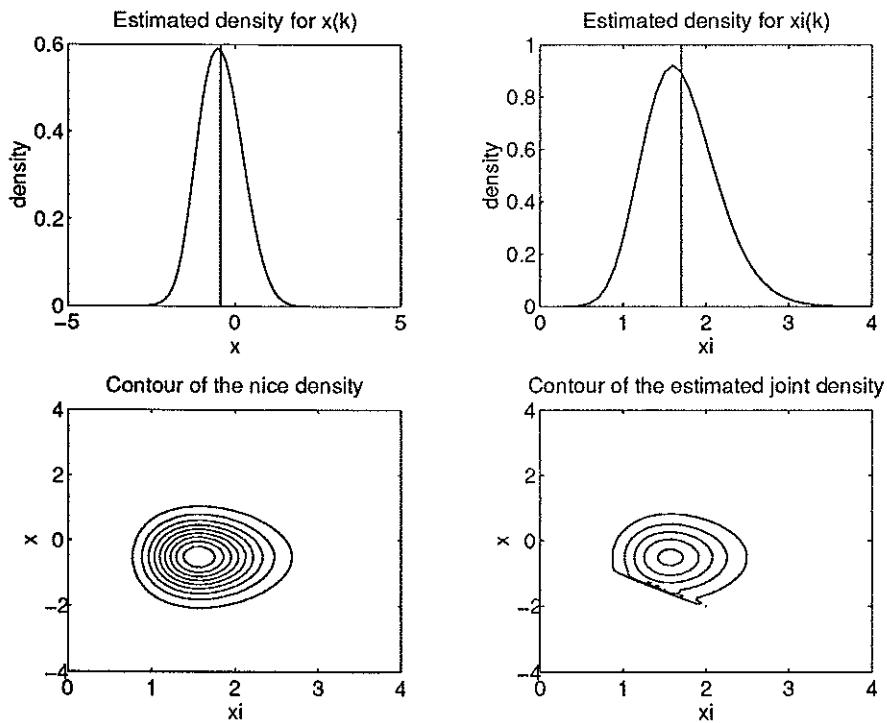
Figur 6.4 Täthetsfunktioner vid tiden 2 baserat på mätningarna. De övre figurerna visar tätheterna för $x(k)$ och $\xi(k)$. Den nedre högra visar nivåkurvor till den simultana täthetsfunktionen mellan $x(k)$ och $\xi(k)$. Den nedre vänstra förklaras i texten.



Figur 6.5 Täthetsfunktioner vid tiden 3 baserat på mätningarna. De övre figurerna visar tätheterna för $x(k)$ och $\xi(k)$. Den nedre högra visar nivåkurvor till den simultana täthetsfunktionen mellan $x(k)$ och $\xi(k)$. Den nedre vänstra förklaras i texten.



Figur 6.6 Täthetsfunktioner vid tiden 6 baserat på mätningarna. De övre figurerna visar tätheterna för $x(k)$ och $\xi(k)$. Den nedre högra visar nivåkurvor till den simultana täthetsfunktionen mellan $x(k)$ och $\xi(k)$. Den nedre vänstra förklaras i texten.



Figur 6.7 Täthetsfunktioner vid tiden 8 baserat på mätningarna. De övre figurerna visar tätheterna för $x(k)$ och $\xi(k)$. Den nedre högra visar nivåkurvor till den simultana täthetsfunktionen mellan $x(k)$ och $\xi(k)$. Den nedre vänstra förklaras i texten.

7. Slutsatser

I denna rapport har olinjär stokastisk reglering av extremvärdet i diskret tid studerats. En olinjär styrlag har tagits fram genom att numeriskt lösa Bellmans ekvation för ett exempel med fullständig tillståndsinformation. Målet med regleringen i detta exempel är att minimera sannolikheten att ett maximum överskriden en given gräns. Simuleringar har gjorts för att jämföra den optimala regulatorn med approximativa regulatorer. Slutsatsen för exemplet blir att de approximativa regulatorerna fungerar i princip lika bra för det beskrivna kriteriet. Dock är det så att minimaluppkorsningsregulatorerna och den optimala regulatorn får mindre styrsignaler. Det finns dock ett fall då den optimala regulatorn blir signifikant bättre och det är när kraven är mycket hårda och maximumet skall hållas innanför gränserna under en kort tid. Hårda krav är mycket brus eller snäva gränser.

Även ofullständig tillståndsinformation har studerats. Det visar sig att att det finns en del problem som behöver tänkas igenom innan man kan lösa Bellmans ekvation. Ett centralt problem är att parametrisera de tätheter för tillståndet som kan tänkas uppstå då den optimala regulatorn reglerar systemet. Denna rapport visar vilken typ av tillståndstäheter som uppstår för ett speciellt system.

Fortsatt arbete

Vid formuleringen av det exempel som skulle lösas numeriskt verkade experiment tyda på att man förutom minst födröjningen två också var tvungen att införa färgat brus. Om det verkligen är så och i så fall varför är något som inte utretts.

När det gäller exemplet har lite olika delproblem utretts som eventuellt skulle kunna sättas samman på en bättre sätt. Detta har ännu inte kunnat slutföras.

Vid ofullständig tillståndsinformation måste man veta vilka tätheter som kommer att uppstå och lagra förluster och styrsignaler för dessa. Det problem som kvarstår här är att finna en parametrering av täheterna som kan uppstå. Ett alternativ kan vara att diskretisera de möjliga utsignalerna istället. Detta är bara lämpligt om minimeringen bara skall ske över ett litet antal sampel eller om styrlagen konvergerar snabbt så att den stationära styrlagen erhålls genom att utföra minimeringen över ett fåtal sampel.

Referenser

- ÅSTRÖM, K. J. (1970): *Introduction to Stochastic Control Theory*. Academic Press, Inc., San Diego.
- ÅSTRÖM, K. J. (1977): "Stochastic control problems." Technical Report TFRT-3147, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.
- ÅSTRÖM, K. J. och A. HELMERSSON (1986): "Dual control of an integrator with unknown gain." *Comp. and Maths. with Appl.*, **12A**, No 6, pp. 653–662.
- ÅSTRÖM, K. J. och B. WITTENMARK (1990): *Computer Controlled Systems—Theory and Design*. Prentice-Hall, Englewood Cliffs, New Jersey.
- BORISSON, U. och R. SYDING (1976): "Self-tuning control of an ore crusher." *Automatica*, **12**, pp. 1–7.
- DAHLEH, M. A. och J. B. PEARSON (1987): " l^1 -optimal feedback controllers for mimo discrete-time systems." *IEEE Transactions on Automatic Control*, **32**, pp. 314–322.
- HANSSON, A. (1991): "Control of extremes and level-crossings in stationary gaussian random processes." I *IEEE Conference on Decision and Control*.
- HANSSON, A. (1992): "Minimum risk control." Technical Report TFRT-3210, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden. Licentiate Thesis.
- HANSSON, A. (1993): "Non-linear stochastic control of critical processes." Technical Report TFRT-7503, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden. Internal report.
- HANSSON, A. och L. NIELSEN (1991): "Control and supervision in sensor-based robotics." I *Proceedings—Robotikdagar—Robotteknik och Verkstadsteknisk Automation—Mot ökad autonomi*, pp. C7-1–10, S-581 83 Linköping, Sweden. Tekniska Högskolan i Linköping.
- HELMERSSON, A. (1981): "Dual reglering." Technical Report TFRT-3245, Institutionen för Reglerteknik Lunds Tekniska Högskola, Lund. Examensarbete.
- KERNIGHAN, B. W. och D. M. RITCHIE (1978): *The C programming language*. Prentice-Hall, Englewood Cliffs, New Jersey.
- LIU, G. P. och V. ZAKIAN (1990): "Sup regulators." I *IEEE Conference on Decision and Control*.
- MATLAB (1992a): *MATLAB External Interface Guide*. the Math Works Inc., Natic, Mass.
- MATLAB (1992b): *MATLAB Reference Guide*. the Math Works Inc., Natic, Mass.
- MATTSSON, S. E. (1984): "Modelling and control of large horizontal axis wind power plants." Technical Report TFRT-1026, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden. Doctoral Dissertation.
- PRESS, W. H., E. A. (1987): *Numerical recipes*. Cambridge University Press, Cambridge.
- SHINSKEY, F. G. (1967): *Process-Control Systems*. McGraw-Hill, Inc., New York.

VIDYASAGAR, M. (1986): "Optimal rejection of persistent bounded disturbances." *IEEE Transactions on Automatic Control*, **31**, pp. 527-534.

A. Ett bevis

Sats

För systemet i avsnitt 4.4 gäller att V kan skrivas som:

$$V(k, x(k), \xi(k)) = W(k, x(k), \theta(\xi(k) - \xi_0)), \quad 0 \leq k \leq N$$

bevis Notera först att

$$V(N, x(N), \xi(N)) = \theta(\xi(N) - \xi_0)$$

vilket innebär att satsen gäller för $k = N$. Antag nu att satsen gäller för $k + 1$ d.v.s. att

$$V(k+1, x(k+1), \xi(k+1)) = W(k+1, x(k+1), \theta(\xi(k+1) - \xi_0))$$

Enligt Bellmans ekvation gäller

$$V(k, x(k), \xi(k)) = \min_{u(k)} \int p_{w(k)}(w) \cdot V(k+1, x(k+1), \xi(k+1)) dw$$

varför det under antagandet följer att

$$V(k, x(k), \xi(k)) = \min_{u(k)} \int p_{w(k)}(w) \cdot W(k+1, x(k+1), \theta(\xi(k+1) - \xi_0)) dw$$

Systemekvationerna ger att

- $x(k+1)$ bestäms av $x(k), u(k)$ samt $w(k)$
- Eftersom $\xi(k+1) = \max(\xi(k), |C_2 x(k+1)|)$ följer att

- $\theta(\xi(k+1) - \xi_0)$ bestäms av $x(k), u(k), w(k)$ samt $\theta(\xi(k) - \xi_0)$

Detta innebär att

$$V(k, x(k), \xi(k)) = W(k, x(k), \theta(\xi(k) - \xi_0))$$

Satsen följer nu av induktion.

Följdsats

För systemet i avsnitt 4.4 gäller att u kan skrivas som

$$u(k, x(k), \xi(k)) = v(k, x(k), \theta(\xi(k) - \xi_0)), \quad 0 \leq k \leq N$$

där u definieras ur den omvänta likheten.

bevis Styrsignalen u bestäms genom att utföra minimeringen

$$\min_{u(k)} \int p_{w(k)}(w) \cdot V(k+1, x(k+1), \xi(k+1)) dw$$

Enligt satsen ovan gäller att V bara beror av $\theta(\xi(k) - \xi_0)$ då det gäller $\xi(k)$. Detta innebär att integralens argument och därmed hela integralen bara har detta ξ -beroende. Styrsignalen kan därför också bara ha detta beroende.

B. Storheter och beteckningar

beteckning	storhet	kallas även
$x(k)$	tillstånd	
$\xi(k)$	maximum hittills	maximum
$[x^T(k) \ \xi(k)]^T$	utökat tillstånd	
J	väntad förlust	förlust
V	minimal delförlust	
$\gamma(k)$	given information	
N	minimeringshorisont	
p	täthetsfunktion	

C. Programlistningar

Numerisk lösning av Bellmans ekvation

- Update.m
- Integrate.c
- InitK.m
- GetU.c
- GetV.c

Simulering av styrlagar

- Simu.m

Linjär minimaluppkorsningsregulator

- intensityd.m
- reguld.m
- LQregul.m

Regulator utan att lösa Bellmans ekvation

- Facit.c

Filtersimulering

- Maxabsfilter.m
- MakePlot.m

Update.m

```

function [out1,out2,out3]=Update(inp);
%
% Generates U,V tables
%
% [V1,U1,V0]=Update(lev); Produces the first tables.
% lev is the table size
%
% [Vlast,Ulast]=Update(V); Produces new tables based on an old V
%

Nu=64;           % Number of different u values
umax=4;          % Maximum u value to use
xi0=1;           % xi0 value

if length(inp)==1      % Make the first V-table of size lev
    lev=inp;
    V=zeros(2*lev-1,lev);
else                  % Decide the table size
    V=inp;
    lev=min(size(V));
end

K=InitK(V);          % Make a coefficient matrix for V
for x1pos=1:2*lev-1  % Go through the s-koordinates
    xT1=(x1pos-lev)/lev; % Decide s_N value
    x1 =xi0*xT1;         % Decide x1   value
    for x2pos=1:lev       % Go through the t-positions
        xT2=(x2pos-1)/lev; % Decide t_N value
        if xT2<0.5          % Decide x2   value
            x2=xi0*sqrt(2*xT2)-x1;
        else
            x2=xi0*0.75/(1-xT2*xT2)-x1;
        end
        x=[x1;x2];

        if length(inp)==1      % Decide Vi value
            Vnow=Integrate(K,x,0);
            u=0;
        else                  % Decide V_n+1 value
            for k=(1-Nu):(Nu-1) % Make the integration for
                uT=k/Nu;          % different u values
                u =uT*umax;
                I(k+Nu)=Integrate(K,x,u);
            end
            [Vnow upos]=min(I);
            uT=(upos-Nu)/Nu;
            u =uT*umax;          % best u value sofar
            if upos>1 & upos<Nu
                uTprev=(upos-1-Nu)/Nu;
                uTnext=(upos+1-Nu)/Nu;
                uprev =uTprev*umax; % nearby u values
                unext =uTnext*umax;
                Vprev =I(upos-1); % nearby V values
                Vnext =I(upos+1);
                p=polyfit([uprev u unext],[Vprev Vnow Vnext],2); % quadratic form
                r=roots(polyder(p));                                % extreme values
                uopt=r(r==real(r) & r<unext & r>uprev);          % possible optimum
                if length(uopt) ~= 0
                    Vopt=Integrate(K,x,uopt);                      % corresponding V
                    if Vopt<Vnow

```

```

        Vnow=Vopt;
        u    =uopt;
    end
end
end
end
Vnow;
Vlast(x1pos,x2pos)=Vnow; % store the best V value for [x1,x2]
U(x1pos,x2pos)=u;         % store the best u value for [x1,x2]
end
end

if length(inp)==1 % return [V1,U1,V0] tables
out1=Vlast;
out2=U;
out3=V;
else             % return [V_n+1,U_n+1] tables
out1=Vlast;
out2=U;
end

```

Integrate.c

```
/* ----- */
/*          */
/* Integration in the full information case      3/7-93 */
/*          */
/* Use I=Integrate(K,[x1;x2],u);                */
/*          */
/* ----- */

/* ----- */
/*           SETUPS */
/* ----- */

#include <math.h>
#include "mex.h"

/* -----
/* design definitions */
/* ----- */

#define eps      0.0001 /* relative error when integrating */
#define sigmadist 5      /* integration width [sigmav] */
#define hmax     0.2      /* max h to accept in integration */
#define xi0      1        /* limit for xi */
#define sigmav   0.5      /* sigma for system noise */
#define c        0.95     /* system parameter */

/* -----
/* usefull definitions */
/* ----- */

#define or          ||
#define and         &&
#define do          do
#define then        then
#define begin       {
#define end         }
#define max(A, B) ((A) > (B) ? (A) : (B))
#define min(A, B) ((A) < (B) ? (A) : (B))
#define abs(x)      sqrt(x*x)
#define pi 3.14159265

/* ----- */
/*           COMPUTATIONAL ROUTINES */
/* ----- */

double GetV(double K[], int lev,
            double x1 , double x2)

/* Calculates an interpolated/extrapolated value of V */

begin
    double xT1,xT2,V;
    int x1pos,x2pos,pos;

/* Using prerequisites for K table and system symmetri */
```

```

if (x1+x2<0) then
begin
  x1=-x1;
  x2=-x2;
end

if (abs(x1)>xi0) then
  return(1);
else
begin

  /* make transformations */

  xT1=x1/xi0;

  if (x2+x1>xi0) then
    xT2=sqrt(1-0.75*xi0/(x1+x2));
  else
    xT2=0.5*(x2+x1)*(x2+x1)/xi0/xi0;

  /* determine nearest possible position to use */

  x1pos=xT1*lev+lev;
  x2pos=xT2*lev+1;

  if (x1pos<=1) then x1pos=2;
  if (x2pos<=1) then x2pos=2;

  if (x1pos>=2*lev-1) then x1pos=2*lev-2;
  if (x2pos>=lev) then x2pos=lev-1;

  /* calculation using a polynomial around the position */

  pos=(x1pos-2)*6
      +(x2pos-2)*6*(2*lev-3);
  V=
    K[pos] + x1* K[pos+1] + x2* K[pos+2]
    + x1*x2*K[pos+3] + x1*x1*K[pos+4] + x2*x2*K[pos+5];
  V=max(V,0);
  V=min(V,1);
  return(V);

end
end

double IntArg(double v,
              double x1given, double x2given,
              double u,
              double K[],     unsigned int lev)

/* Gives the argument in the integration */
/* That is p(w)*V */

begin
  double x1,x2,xi,P0,P1,dPdv;

  /* system definition, should not be changed */

  x1= x1given + x2given +c*v;
  x2=      + x2given +v +u;

  /* density for noise v */

```

```

dPdv=1.0/sigmay/sqrt(2*pi)*exp(-v*v/2/(sigmay*sigmay));

return(dPdv*GetV(K,lev,xi,x2));
end

double DoIntegrate(double x1given, double x2given,
                    double u,          double K[],
                    unsigned int lev)

/* integrating using simpson and Richardson extrapolation */
/* results in E(V(k+1)|x(k),xi(k)) */

begin
    double diffkv,dist,h,sum,sum4,Isimp1,Isimp2,I1,I2;
    int     Nnew,fnr,k;

    diffkv=1.0;
    Nnew=1;
    dist=sigmay*sigmadist;                                /* v at "infinity" */
    h=dist;                                              /* First integration step */
    sum     =IntArg(-dist,x1given,x2given,u,K,lev)+
              IntArg( dist,x1given,x2given,u,K,lev);
    sum4   =IntArg( 0.0 ,x1given,x2given,u,K,lev);
    Isimp1 =h/3*(sum+4.0*sum4);
    I1     =Isimp1;
    while (diffkv>eps or h>hmax)                      /* Sum until integral converged */
    begin
        h=h/2;                                         /* Halven step */
        sum =sum+2*sum4;
        sum4=0;
        fnr =-1;
        Nnew=Nnew*2;
        for (k=1;k<=Nnew;k++)
        begin
            fnr=fnr+2;
            sum4=sum4+IntArg(-dist+fnr*h,x1given,x2given,u,K,lev);
        end
        Isimp2=h/3*(sum+4.0*sum4);
        I2=(16*Isimp2-Isimp1)/15;           /* Extrapolation */
        if (abs(I2)>eps) then             /* Check for nonzero sum value */
            diffkv=abs((I2-I1)/I2);       /* Relative error */
        else
            diffkv=0.0;
        Isimp1=Isimp2;
        I1     =I2;
    end
    return(I1);
end

/* -----
   *          GATEWAY ROUTINE
   * -----
   */

void mexFunction(int nlhs,
                 Matrix *plhs[],
                 int nrhs,
                 Matrix      *prhs[])
begin
    unsigned int m,n;
    double *Y,*K,*X,*U;

```

```

/* -----
/* Check for proper number of arguments */
/* ----- */

if (nrhs !=3) then
    mexErrMsgTxt("Use I=Integrate(K,x,u)");
else if (nlhs > 1) then
    mexErrMsgTxt( "Only one output argument is possible");

/* -----
/* Check the dimensions of input arguments */
/* ----- */

m=mxGetM(prhs[0]);
n=mxGetN(prhs[0]);
if (!mxIsNumeric(prhs[0]) or mxIsComplex(prhs[0]) or
    !mxIsFull(prhs[0]) or !mxIsDouble(prhs[0])) then
    mexErrMsgTxt("K matrix doesn't fitt");
if (!mxIsNumeric(prhs[1]) or mxIsComplex(prhs[1]) or
    !mxIsFull(prhs[1]) or !mxIsDouble(prhs[1]) or
    (mxGetM(prhs[1]) != 2)or (mxGetN(prhs[1]) != 1)) then
    mexErrMsgTxt("X matrix doesn't fitt");
if (!mxIsNumeric(prhs[2]) or mxIsComplex(prhs[2]) or
    !mxIsFull(prhs[2]) or !mxIsDouble(prhs[2]) or
    (mxGetM(prhs[2]) != 1)or (mxGetN(prhs[2]) != 1)) then
    mexErrMsgTxt("U matrix doesn't fitt");

/* -----
/* Create a matrix for the return argument */
/* ----- */

plhs[0]=mxCreateFull(1,1, REAL);

/* -----
/* Assign pointers to the various parameters */
/* ----- */

Y= mxGetPr(plhs[0]);
K= mxGetPr(prhs[0]);
X= mxGetPr(prhs[1]);
U= mxGetPr(prhs[2]);

/* -----
/* Do the actual computations in a subroutine */
/* ----- */

Y[0]=DoIntegrate(X[0],X[1],U[0],K,n+2);
end

```

InitK.m

```
function K=InitK(V);
%
% K=InitK(V);
% C=InitK(U);
% the coefficients for interpolation in V or U is returned
%
% a quadratic form is made using nine values and LS-method
% V or U must be 3x3 or bigger

xi0=1;      % xi0 value

lev=min(size(V));                         % decide table size
for x1pos=2:(2*lev-2)                     % go through all table positions
    for x2pos=2:(lev-1)
        Fi=[];
        Y=[];
        for i=-2:0                          % use nearby values of s
            xT1=(i+x1pos-lev+1)/lev;       % decide s_N values
            x1=xi0*xT1;                   % decide x1-values
            for j=-2:0                      % use nearby values of t
                xT2=(j+x2pos)/lev;         % decide t_N values
                if xT2<0.5                % decide x2 values
                    x2=xi0*sqrt(2*xT2)-x1;
                else
                    x2=xi0*3/4/(1-xT2*xT2)-x1;
                end
                Fi=[Fi ; 1 x1 x2 x1*x2 x1*x1 x2*x2]; % store quadratic values
                Y =[Y ; V(x1pos+i+1,x2pos+j+1)];      % store V values
            end
        end
        Theta=Fi\Y;                         % Make a least square estimation
        K(((x1pos-2)*6+1):(x1pos*6-6),x2pos-1)=Theta; % Store coefficients
    end
end
```

GetU.c

```
/*
/*
/* Interpolation in table U      27/6/93
/*
/* Use U=GetU(C,[x1;x2]);
/*
/*
/* -----
*/
/*
SETUPS
/*
/*
#include <math.h>
#include "mex.h"

/*
/* design definitions */
/*
#define xi0      1

/*
/* useful definitions */
/*
#define or      ||
#define and     &&
#define do      do
#define then    while
#define begin   {
#define end     }
#define max(A, B) ((A) > (B) ? (A) : (B))
#define min(A, B) ((A) < (B) ? (A) : (B))
#define abs(x)   sqrt(x*x)
#define pi 3.14159265

/*
COMPUTATIONAL ROUTINES
/*
double GetV(double K[], int lev,
            double x1 , double x2)
begin
    double xT1,xT2,V;
    int x1pos,x2pos,pos;

    /* Using symmetri knowledge */

    if (x1+x2<0) then
        return(-GetV(K,lev,-x1,-x2));

    begin
        /* make transformations */

```

```

xT1=xi1/xi0;

if (x2+x1>xi0) then
  xT2=sqrt(1-0.75*xi0/(x1+x2));
else
  xT2=0.5*(x2+x1)*(x2+x1)/xi0/xi0;

/* determine nearest possible position to use */

x1pos=xT1*lev+lev;
x2pos=xT2*lev+1;

if (x1pos<=1) then x1pos=2;
if (x2pos<=1) then x2pos=2;

if (x1pos>=2*lev-1) then x1pos=2*lev-2;
if (x2pos>=lev) then x2pos=lev-1;

/* calculation using a polynomial around the position */

pos=(x1pos-2)*6
  +(x2pos-2)*6*(2*lev-3);
V=
  K[pos] + x1* K[pos+1] + x2* K[pos+2]
  + x1*x2*K[pos+3] + x1*x1*K[pos+4] + x2*x2*K[pos+5] ;

return(V);
end
end
/* -----
/*          GATEWAY ROUTINE
/* -----
*/
void mexFunction(int nlhs,
                 Matrix *plhs[],
                 int nrhs,
                 Matrix *prhs[])
begin
  unsigned int m,n;
  double *Y,*K,*X;
  int i;

/* -----
/* Check for proper number of arguments */
/* ----- */

if (nrhs !=2) then
  mexErrMsgTxt("Use V=GetV(K,[x1;x2]);");
else if (nlhs > 1) then
  mexErrMsgTxt( "Only one output argument is possible");

/* -----
/* Check the dimensions of input arguments */
/* ----- */

m=mxGetM(prhs[0]);
n=mxGetN(prhs[0]);
if (!mxIsNumeric(prhs[0]) or mxIsComplex(prhs[0]) or
    !mxIsFull(prhs[0]) or !mxIsDouble(prhs[0])) then
  mexErrMsgTxt("K matrix doesn't fit");

/* -----

```

```
/* Create a matrix for the return argument */
/* ----- */

plhs[0]=mxCreateFull(1,1, REAL);

/* -----
/* Assign pointers to the various parameters */
/* ----- */

Y= mxGetPr(plhs[0]);
K= mxGetPr(prhs[0]);
X= mxGetPr(prhs[1]);

/* -----
/* Do the actual computations in a subroutine */
/* ----- */

Y[0]=GetV(K,n+2,X[0],X[1]);
end
```

GetV.c

```
/* ----- */
/*          */
/*   Interpolation in table V      27/6-93      */
/*          */
/*   Use V=GetV(K,[x1;x2]);      */
/*          */
/* ----- */

/* ----- */
/*           SETUPS      */
/* ----- */

#include <math.h>
#include "mex.h"

/* ----- */
/* design definitions */
/* ----- */

#define xi0      1

/* ----- */
/* usefull definitions */
/* ----- */

#define or      ||
#define and     &&
#define do      do
#define then    then
#define begin   {
#define end     }
#define max(A, B) ((A) > (B) ? (A) : (B))
#define min(A, B) ((A) < (B) ? (A) : (B))
#define abs(x)   sqrt(x*x)
#define pi 3.14159265

/* ----- */
/*           COMPUTATIONAL ROUTINES      */
/* ----- */

double GetV(double K[], int lev,
            double x1 , double x2)
begin
    double xT1,xT2,V;
    int x1pos,x2pos,pos;

    /* use symmetry */

    if (x1+x2<0) then
    begin
        x1=-x1;
        x2=-x2;
    end

    /* use preknowledge */
```

```

if (abs(x1)>xi0) then
    return(1);
else
begin

/* make transformations */

xT1=x1/xi0;

if (x2+x1>xi0) then
    xT2=sqrt(1-0.75*xi0/(x1+x2));
else
    xT2=0.5*(x2+x1)*(x2+x1)/xi0/xi0;

/* determine nearest possible position to use */

x1pos=xT1*lev+lev;
x2pos=xT2*lev+i;

if (x1pos<=1) then x1pos=2;
if (x2pos<=1) then x2pos=2;

if (x1pos>=2*lev-1) then x1pos=2*lev-2;
if (x2pos>=lev) then x2pos=lev-1;

/* calculation using a polynomial around the position */

pos=(x1pos-2)*6
    +(x2pos-2)*6*(2*lev-3);
V=
    K[pos] + x1* K[pos+1] + x2* K[pos+2]
    + x1*x2*K[pos+3] + x1*x1*K[pos+4] + x2*x2*K[pos+5] ;

return(V);
end
end
/*
*----- */          */
/*           GATEWAY ROUTINE          */
/*----- */          */
void mexFunction(int nlhs,
                 Matrix *plhs[],
                 int nrhs,
                 Matrix *prhs[])
begin
    unsigned int m,n;
    double *Y,*K,*X;
    int i;

/* ----- */
/* Check for proper number of arguments */
/* ----- */

if (nrhs !=2) then
    mexErrMsgTxt("Use V=GetV(K,[x1;x2]);");
else if (nlhs > 1) then
    mexErrMsgTxt( "Only one output argument is possible");

/* ----- */
/* Check the dimensions of input arguments */
/* ----- */

```

```

m=mxGetM(prhs[0]);
n=mxGetN(prhs[0]);
if (!mxIsNumeric(prhs[0]) or mxIsComplex(prhs[0]) or
    !mxIsFull(prhs[0]) or !mxIsDouble(prhs[0])) then
    mexErrMsgTxt("K matrix doesn't fitt");

/* -----
/* Create a matrix for the return argument */
/* ----- */

plhs[0]=mxCreateFull(1,1, REAL);

/* -----
/* Assign pointers to the various parameters */
/* ----- */

Y= mxGetPr(plhs[0]);
K= mxGetPr(prhs[0]);
X= mxGetPr(prhs[1]);

/* -----
/* Do the actual computations in a subroutine */
/* ----- */

Y[0]=GetV(K,n+2,X[0],X[1]);
end

```

Simu.m

```
function [J,y,u]=Simu(Q,x,iter,N,Cstat)

% [J,y,u]=Simu(Q,x,iter) simulates the system over two samples
%
% Q specifies the controller. It can be a controller given by C or
% it can be a linear feedback given by L
% x is the initial state for each simulation
% iter is the number of simulations
%
% [J,y,u]=Simu(Q,x,iter,N,Cstat) simulates over N samples
%
% When Q=C then Cstat specifies the controller for all times except the last
%
% NOTE: The systemparameters must be changed IN this m-file

sigmaw=0.2; % system parameters
c=0.7;

if nargin < 3
    iter=1;
end

if nargin < 4
    N=2;
end

u=[];
y=[];
J=[];

for j=1:iter % make iter simulations
    x1=x(1); % set initial state
    x2=x(2);
    xi=0; % initialize xi to zero
    ut=[];
    yt=[];
    for k=1:N % simulate over N samples
        w =randn*sigmaw; % system noise
        if size(Q)==[1 2] % linear control law
            uc =-Q*[x1;x2];
        else % optimal control law
            if N-k>0 & k~=N-1
                uc=GetU(Cstat,[x1;x2]); % optimal stationary control
            else
                uc=GetU(Q,[x1,x2]); % optimal first sample control
            end
        end
        x1=x1+x2+c*w; % system update
        x2=x2+w+uc;
        xi=max(xi,abs(x1));
        ut=[ut; uc];
        yt=[yt; x1];
    end
    u=[u ut]; % store control
    y=[y yt]; % store state x1
    J=[J xi>1]; % store lossvalue
end
```

intensityd.m

```
function [int, valpha, vbeta, vz, vu, rho] = intensityd(A,B1,B2,C1,C2,D1,D2, ...
R1,R2,R3,R12,R23,z0,mz,start,step,stop)
%
% [int, valpha, vbeta, vz, vu, rho] = intensityd(A,B1,B2,C1,C2,D1,D2, ...
% R1,R2,R3,R12,R23,z0,mz,start,step,stop)
%
%
% the upcrossing intensity int is calculated for different
% controllers specified by rho=start:step:stop
% some variances are also determined
%
%
res = 25;
nstdev = 6;
u = z0-mz;
int = [];
valpha = [];
vbeta = [];
vz = [];
vu = [];
rho = [];
for rh=start:step:stop % calculate the upcrossingintensity for different rho
    [n,d,valphai,vbetai,vz1,vui]=reguld(A,B1,B2,C1,C2,D1,D2,R1, ...
R2,R3,R12,R23,rh); % calculates variances
    int1 = 0;
    sa = sqrt(valphai);
    sb = sqrt(vbetai);
    limit = nstdev*max(sa,sb);
    h = min(sa,sb)/res; % integration step
    for i=h:h:limit % outer integral
        xl=(2*u-(i-h/2))/sa; % inner integral limits
        xu=(2*u+i-h/2) /sa;
        if xu>xl,
            int1 = int1+h/sb/sqrt(2*pi)*exp(-(i-h/2)^2/2/(sb^2))*...
            0.5*(erf(xu/sqrt(2))-erf(xl/sqrt(2))); % inner integral
        end
    end
    int = [int int1]; % store uppcrossing intensity
    valpha = [valpha valphai]; % store variances
    vbeta = [vbeta vbetai];
    vz = [vz vz1];
    vu = [vu vui];
    rho = [rho rh];
end
```

reguld.m

```
function [n,d,valpha,vbeta,vz,vu] = reguld(A,B1,B2,C1,C2,D1,D2, ...
R1,R2,R3,R12,R23,rho)
%
% REGUL Discrete-time regulator U(s) = -n(s)/d(s) * Y(s)
%
% [n,d,valpha,vbeta,vz,vu] = reguld(A,B1,B2,C1,C2,D,R1,R2,R12,rho)
%
% The controller is designed such that the cost function:
%
% J = E {Sum [z'z + rho^2 * (dz/dt)'dz/dt] dt},
%
% is minimized subject to the constraint equation:
%
% dx = A x dt + B1 u dt + B2 dv
% dy = C1 x dt + D1 de
%
% where
%
% z = C2 x + D2 dw
%
% and
%
% E{dv} = E{de} = 0,
% E{dv dv'} = R1 dt, E{de de'} = R2 dt, E{dv de'} = R12 dt
%
% The outputs varz and varderz are the variances of z and the dz/dt
%
% Note that if C2 * B2 is not equal to zero, the loss function J
% will be infinite.
%
% Note that if C2 * B1 is equal to zero, the optimization routine
% will be singular. This problem is solved by adding epsilon * u'u
% to the loss function, where epsilon should be a small number such as
% 10^(-14).
%
% Sufficient conditions for the existence of a solution to the
% problem are that C1 (sI - A)^(-1) B2 has full rank and R1 is non-
% singular with rank greater than or equal to the number of outputs,
% and that D2 * R2 * D2' is non-singular. The last condi-
% tion can be weakened, see Shaked, U., A general transfer-function
% approach to the steady-state linear quadratic Gaussian Stochastic
% control problem, Int. J. Control., 1976, vol. 24, No. 6, pp 788--790.
% Note that if the first condition is not fulfilled, it may sometimes
% be possible to choose equivalent B2 and R1, such that it is fulfilled.
%
if R2==0 & R3==0 % The special case in this report
    [L,valpha,vbeta,vz,vu] = LQregul(A,B1,B2,C2,R1,rho);
else
%
nn=size(A);
nn=nn(1:1);
Q1 = C2'*C2+A'*C2'*C2*A+(1-2*rho)*(A'*C2'*C2+C2'*C2*A);
Q12 = A'*C2'*C2*B1+(1-2*rho)*C2'*C2*B1;
Q2 = B1'*C2'*C2*B1;
Q23 = B1'*C2'*C2*B2;
Q24 = (1-2*rho)*B1'*C2'*D2;
%[L,Lv,lr,S] = lqr(A,B1,C1,Q1,Q2,Q12);
S = dare(A,B1,Q1,Q2,Q12);
barQ = (Q2 + B1'*S*B1);
L = barQ\((B1'*S*A + Q12'));

```

```

Lv = barQ\B1'*S*B2 + Q23);
Lw = barQ\Q24;
[K,Kf,Kv,P,Pf] = lqed(A,C1,B2*R1*B2',D1*R2*D1',B2*R12*D1');
P = dare(A',C1',B2*R1*B2',D1*R2*D1',B2*R12*D1');
Ry = C1*P*C1' + D1*R2*D1';
Kf = P*C1'/Ry;
Kv = R12*D1'/Ry;
K = A*Kf+B2*Kv;
Kw = R23*D1'/Ry;
Ly = L*Kf+Lv*Kv+Lw*Kw;
Lx = L-Ly*C1;
Kx = K-B1*Ly;
Ac = A-B1*Lx-K*C1;
Bc = Kx;
Cc = Lx;
Dc = Ly;
[n,d] = ss2tf(Ac,Bc,Cc,Dc,1);
Aclosed = [A-B1*L B1*Lx;zeros(size(A)) A-K*C1];
factor = [B2 -B1*Ly*D1;B2 -K*D1];
Rclosed = factor*[R1 R12;R12' R2]*factor';
P = dlyap(Aclosed,Rclosed);
valpha = [C2 zeros(size(C2))]*((Aclosed+eye(size(Aclosed)))*...
P*(Aclosed+eye(size(Aclosed))')+Rclosed)*[C2 zeros(size(C2))]'*2*D2*R3*D2';
vbeta = [C2 zeros(size(C2))]*((Aclosed-eye(size(Aclosed)))*...
P*(Aclosed-eye(size(Aclosed))')+Rclosed)*[C2 zeros(size(C2))]'*2*D2*R3*D2';
vz = [C2 zeros(size(C2))]*P*[C2 zeros(size(C2))]'*D2*R3*D2';
vu = [-L Lx]*P*[-L Lx]'*Ly*D1*R2*(Ly*D1)';
%
end

```

LQregul.m

```
function [L,valpha,vbeta,vz,vu] = LQregul(A,B1,B2,C2,R1,rho)
%
% [L,valpha,vbeta,vz,vu] = LQregul(A,B1,B2,C2,R1,rho)
%
% the regulator u = -Lx is designed
% L and the variances are left as results
%
Q1 = C2'*C2+A'*C2'*C2*A+(1-2*rho)*(A'*C2'*C2+C2'*C2*A); % LQ matrixes
Q12 = A'*C2'*C2*B1+(1-2*rho)*C2'*C2*B1;
Q2 = B1'*C2'*C2*B1;

S = dare(A,B1,Q1,Q2,Q12); % solution of LQ problem
barQ = (Q2 + B1'*S*B1);
L = barQ\((B1'*S*A + Q12');

P = dlyap(A-B1*L,B2*R1*B2'); % variance matrix

vz = C2*P*C2'; % variance for z
vu = L*P*L'; % variance for u
valpha = 2*C2*(eye(size(A))+A-B1*L)*P*C2'; % variance for alpha
vbeta = 2*C2*(eye(size(A))-A+B1*L)*P*C2'; % variance for beta
```

Facit.c

```
/* ----- */  
/* */  
/* Full info */  
/* 21/7-93 */  
/* */  
/* Use Use [mean,norm]=Facit([x1;x2]); */  
/* */  
/* ----- */  
  
/* ----- */  
/* SETUPS */  
/* ----- */  
  
#include <math.h>  
#include "mex.h"  
  
/* ----- */  
/* design definitions */  
/* ----- */  
  
#define eps 0.01 /* relative error when integrating */  
#define hmax 0.2 /* max h to accept in integration */  
#define dist 5 /* integration width [sigma] */  
#define xi0 1 /* limit for xi */  
#define c 0.7 /* system koefficient */  
#define sigmaw 0.2 /* system noise level */  
  
/* ----- */  
/* usefull definitions */  
/* ----- */  
  
#define or ||  
#define and &&  
#define do  
#define then  
#define begin {  
#define end }  
#define max(A, B) ((A) > (B) ? (A) : (B))  
#define min(A, B) ((A) < (B) ? (A) : (B))  
#define abs(x) sqrt(x*x)  
#define pi 3.141592653  
  
/* ----- */  
/* COMPUTATIONAL ROUTINES */  
/* ----- */  
  
double Fi(double w)  
/* Gaussian density for noise w */  
begin  
    return(1/sqrt(2*pi)/sigmaw*exp(-0.5*w*w/(sigmaw*sigmaw)));  
end  
  
  
double SingIntegral(double wstart,double wstop)  
/* P(wstart<w<wstop) is returned */
```

```

/* integrating using simpson and Richardson extrapolation */

begin
  double diffkv,h,sum,sum4,Isimp1,Isimp2,I1,I2;
  int   Nnew,fnr,k ;

  diffkv=1.0;
  Nnew=i;
  h=(wstop-wstart)/2;           /* First integration step */
  sum    =Fi(wstart)+          Fi(wstop);
  sum4   =Fi(wstart+h);
  Isimp1 =h/3*(sum+4.0*sum4);
  I1     =Isimp1;
  while (diffkv>eps or h>hmax) /* Sum until integral converged */
  begin
    h=h/2;                     /* Halven step */
    sum =sum+2*sum4;
    sum4=0;
    fnr =-1;
    Nnew=Nnew*2;
    for (k=1;k<=Nnew;k++)
    begin
      fnr=fnr+2;
      sum4=sum4+Fi(wstart+fnr*h);
    end
    Isimp2=h/3*(sum+4.0*sum4);
    I2=(16*Isimp2-Isimp1)/15;  /* Extrapolation */
    if (abs(I2)>eps) then      /* Check for nonzero sum value */
      diffkv=abs((I2-I1)/I2);  /* relative error */
    else
      diffkv=0.0;
    Isimp1=Isimp2;
    I1   =I2;
  end
  return(I1);
end

```

```

double ProdIntegral(double zeta,double wstart,double wstop)

/* calculates P(eps1<w<eps2)*p(zeta|wstart<w<wstop)      */
/* makes an integration with the product argument            */
/* integrating using simpson and Richardson extrapolation */

begin
  double diffkv,h,sum,sum4,Isimp1,Isimp2,I1,I2;
  int   Nnew,fnr,k ;

  diffkv=1.0;
  Nnew=i;
  h=(wstop-wstart)/2;           /* First integration step */
  sum    =Fi(wstart)*Fi((zeta-(1+c)*wstart)/c)/c+
          Fi(wstop)*Fi((zeta-(1+c)*wstop)/c)/c;
  sum4   =Fi(wstart+h)*Fi((zeta-(1+c)*(wstart+h))/c)/c;
  Isimp1 =h/3*(sum+4.0*sum4);
  I1     =Isimp1;
  while (diffkv>eps or h>hmax) /* Sum until integral converged */
  begin
    h=h/2;                     /* Halven step */

```

```

        sum =sum+2*sum4;
        sum4=0;
        fnr ==-1;
        Nnew=Nnew*2;
        for (k=1;k<=Nnew;k++)
        begin
            fnr=fnr+2;
            sum4=sum4+Fi(wstart+fnr*h)*Fi((zeta-(i+c)*(wstart+fnr*h))/c)/c;
        end
        Isimp2=h/3*(sum+4.0*sum4);
        I2=(16*Isimp2-Isimp1)/15;      /* Extrapolation */
        if (abs(I2)>eps) then          /* Check for nonzero sum value */
            diffkv=abs((I2-I1)/I2);    /* relative error */
        else
            diffkv=0.0;
        Isimp1=Isimp2;
        I1     =I2;
        end
        return(I1);
    end

double DoubleIntegral(double zstart,double zstop,double wstart,double wstop)
/* calculate P(eps1<w<eps2)*E(zeta|eps1<w<eps2) */
/* integrating using simpson and Richardson extrapolation */

begin
    double diffkv,h,sum,sum4,Isimp1,Isimp2,I1,I2;
    int    Nnew,fnr,k ;
    diffkv=1.0;
    Nnew=1;
    h=(zstop-zstart)/2;           /* First integration step */
    sum    =zstart*ProdIntegral(zstart,wstart,wstop)+
            zstop*ProdIntegral(zstop,wstart,wstop);
    sum4   =(zstart+h)*ProdIntegral(zstart+h,wstart,wstop);
    Isimp1 =h/3*(sum+4.0*sum4);
    I1     =Isimp1;
    while (diffkv>eps or h>hmax) /* Sum until integral converged */
    begin
        h=h/2;                      /* Halven step */
        sum =sum+2*sum4;
        sum4=0;
        fnr ==-1;
        Nnew=Nnew*2;
        for (k=1;k<=Nnew;k++)
        begin
            fnr=fnr+2;
            sum4=sum4+(zstart+fnr*h)*ProdIntegral(zstart+fnr*h,wstart,wstop);
        end
        Isimp2=h/3*(sum+4.0*sum4);
        I2=(16*Isimp2-Isimp1)/15;      /* extrapolation */
        if (abs(I2)>eps) then          /* Check for nonzero sum value */
            diffkv=abs((I2-I1)/I2);    /* relative error */
        else
            diffkv=0.0;
        Isimp1=Isimp2;
        I1     =I2;
        end
        return(I1);
    end

```

```

/* ----- */
/*          GATEWAY ROUTINE          */
/* ----- */

void mexFunction(int nlhs,
                 Matrix *plhs[],
                 int nrhs,
                 Matrix      *prhs[])
{
begin
    double *N,*M,*X,x1,x2,eps1,eps2,zeta1,zeta2,I0,I1;

/* ----- */
/* Check for proper number of arguments */
/* ----- */

if (nrhs !=1) then
    mexErrMsgTxt("Use [mean,norm]=Facit([x1;x2]);");
else if (nlhs > 2) then
    mexErrMsgTxt( "Only two output arguments are available");

/* ----- */
/* Check the dimensions of input arguments */
/* ----- */

if (!mxIsNumeric(prhs[0]) or   mxIsComplex(prhs[0]) or
    !mxIsFull(prhs[0])      or   !mxIsDouble(prhs[0])  or
    (mxGetM(prhs[0]) != 2)or (mxGetN(prhs[0]) != 1)) then
    mexErrMsgTxt("X matrix doesn't fit");

/* ----- */
/* Create a matrix for the return argument */
/* ----- */

plhs[0]=mxCreateFull(1,1, REAL);
plhs[1]=mxCreateFull(1,1, REAL);

/* ----- */
/* Assign pointers to the various parameters */
/* ----- */

M= mxGetPr(plhs[0]);
N= mxGetPr(plhs[1]);
X= mxGetPr(prhs[0]);

/* ----- */
/* Do the actual computations in a subroutine */
/* ----- */

x1= X[0];           /* initial states */
x2= X[1];

eps1=(-xi0-x1-x2)/c; /* count epsilon values */
eps2=(xi0-x1-x2)/c;

```

```
zeta1=(1+c)*eps1-c*sigmaw*dist; /* zeta at "-infinity" */
zeta2=(1+c)*eps2+c*sigmaw*dist; /* zeta at "+infinity" */

/* calculate P(eps1<w<eps2)*E(zeta|eps1<w<eps2) */

I1=DoubleIntegral(zeta1,zeta2,eps1,eps2);

/* calculate the probability P(eps1<w<eps2) */

I0=SingIntegral(eps1,eps2);

M[0]=-x1-2*x2-I1/I0; /* controlvalue */
N[0]=I0;
end
```

```
zeta1=(1+c)*eps1-c*sigmaw*dist; /* zeta at "-infinity" */
zeta2=(1+c)*eps2+c*sigmaw*dist; /* zeta at "+infinity" */

/* calculate P(eps1<w<eps2)*E(zeta|eps1<w<eps2) */

I1=DoubleIntegral(zeta1,zeta2,eps1,eps2);

/* calculate the probability P(eps1<w<eps2) */

I0=SingIntegral(eps1,eps2);

M[0]=-x1-2*x2-I1/I0; /* controlvalue */
N[0]=I0;
end
```

Maxabsfilter.m

```
function Maxabsfilter(A,B,C1,C2,m0,R0,R1,R2,N,seed)
%
% function Maxabsfilter(A,B,C1,C2,m0,R0,R1,R2,N,seed)
%
% Solves the nonlinear recursive filtering problem for the estimation of the
% distribution function of the states x and xi in the difference equation
%
%     x(k+1) = A*x(k) + B*u(k) + v(k)
%     xi(k+1) = max(xi(k),|C2*(A*x(k) + B*u(k) + v(k))|)
%     y(k) = C1*x(k) + e(k)
%     z(k) = C2*x(k)
%
% where xi(0) = |C2*x(0)|, and where x(0), v(k) and e(k) are independent
% Gaussian random variables with means m0, 0, and 0 respectively and
% variances R0, R1, and R2 respectively.
%
% Note that the state x is scalar and that there is no cross-covariance
% between v and e. Further it is assumed that u(k) = 0.
%
% The resulting distribution function will be plotted at each time
% instant.
%
% joint distribution function for x and xi
% distribution function for x
% distribution function for xi
%
randn('seed',seed);
xmax = 4;                                % maximum state value in density vectors
xmin = -xmax;
ximin = 0;
ximax = xmax;
xstep = 0.1;                               % distance betwin consecutive state values
xistep = xstep;
sigmav = sqrt(R1);
sigmae = sqrt(R2);
sigma = sqrt(C1*R1*C1'+R2);
%-----
% Initialize the joint distribution for x and xi
%-----
m = [m0; C2*m0];
Q = [R0 R0*C2'; C2*R0 C2*R0*C2']+eps*100*eye(2);
invQ = inv(Q);
pk = [];
for x = xmin:xstep:xmax,
    pkx = [];
    for xi = ximin:xistep:ximax,
        if (xi==abs(C2*x))      % only probability along dirac lines
            pkx=[pkx 1/(2*pi*sqrt(R0))*exp(-(x-m0)'*inv(R0)*(x-m0))];
        else
            pkx=[pkx 0];
        end
    end
    pk = [pk;pkx];                % p(x(0),xi(0))
end
pk = pk/(sum(sum(pk))*xstep*xistep);        % normalize density
xhat = (xmin:xstep:xmax)*sum(pk)''*xistep*xstep;    % mean x
xihat = (ximin:xistep:ximax)*sum(pk)''*xistep*xstep;    % mean xi
MakePlot(-1,pk,xhat,xmin,xstep,xmax,xihat,ximin,xistep,ximax);
%-----
% Initiali the simulation
```

```

%-----
xk = m0+R0*randn; % initiali x
xik = abs(C2*xk); % initiali xi
%-----
% Iterate from time 0 to N
%-----
uk = 0;
for k = 0:N, % simulate over N samples
    k
    if k==0,
        xnew=xk;
        xinew=xik;
    else % update the states
        vk = sigmav*randn;
        xnew = A*xk+B*uk+vk;
        xinew = max([xik abs(C2*(A*xk+B*uk+vk))]);
    end
    ynew = C1*xnew+sigmae*randn; % decide output from system
    pnew = [];
    pnorm = [];
    for r = xmin:xstep:xmax, % make p(x(k+1),xi(k+1)|Y(k+1))
        pxnew = [];
        pxnorm = [];
        num = 1/sigmae*Fi((ynew-C1*x)/sigmae); % p(y(k+1)|x(k+1))
        den = 1/sigma*Fi((ynew-C1*(A*xhat+B*uk))/sigma); % p(y(k+1)|Y(k))
        alpha = num/den;
        for xi = ximin:xistep:ximax, % make p(x(k+1),xi(k+1)|Y(k))
            beta = 0;
            gamma = 0;
            delta = 0;
            for xk = -xi/C2:xstep:xi/C2 % sum nice parts
                diff = 1/sigmav*Fi((x-A*xk-B*uk)/sigmav)* ...
                    pk((xk-xmin)/xstep+1,(xi-ximin)/xistep+1)*xstep;
                if abs(x) <= xi/C2
                    beta=beta+diff; % truncated nice density
                end;
                delta=delta+diff; % nice density
            end

            if abs(x)==xi/C2 % sum dirac lines
                for xik = 0:xistep:xi
                    for xk = -xik/C2:xistep:xik/C2
                        gamma= gamma +1/C2/sigmav*Fi((x-A*xk-B*uk)/sigmav)* ...
                            pk((xk-xmin)/xstep+1,(xik-ximin)/xistep+1)*xstep*xistep;
                    end
                end
            end
            pxnew = [pxnew alpha*(beta+gamma)];
            pxnorm = [pxnorm alpha*delta];
        end
        pnew = [pnew;pxnew]; % p(x(k+1),xi(k+1)|Y(k))
        pnorm = [pnorm;pxnorm]; % nice density
    end
    xk = xnew;
    xik = xinew;
    pk = pnew;
    pk = pk/(sum(pk))*xstep*xistep; % normalize
    pnorm=pnorm/(sum(sum(pnorm))*xstep*xistep); % normalize
    xhat = (xmin:xstep:xmax)*sum(pk)'*xistep*xstep; % mean x
    xihat = (ximin:xistep:ximax)*sum(pk)'*xistep*xstep; % mean xi
    MakePlot(k,pk,xhat,xmin,xstep,xmax,xihat,ximin,xistep,ximax,pnorm);
end

```

MakePlot.m

```
function MakePlot(k,pk,xhat,xmin,xstep,xmax,xihat,ximin,xistep,ximax,pnorm)

% Make density figures

toprinter=1; % store the plots on files

clf;

% plot density for x

subplot(221);
hold off;
plot(xmin:xstep:xmax,sum(pk')*xistep);
axis(axis);
hold on;
plot([xhat xhat],[0 10]);
xlabel('x');
ylabel('density');
title('Estimated density for x(k)');

% plot density for xi

subplot(222);
hold off;
plot(ximin:xistep:ximax,sum(pk)*xstep);
axis(axis);
hold on;
plot([xihat xihat],[0 10]);
xlabel('xi');
ylabel('density');
title('Estimated density for xi(k)');

% plot contour of the nice density

if k>=0
    subplot(223);
    contour(ximin:xistep:ximax,xmin:xstep:xmax,pnorm);
    xlabel('xi');
    ylabel('x');
    title('Contour of the nice density');
end

% plot contour of joint density

subplot(224);
contour(ximin:xistep:ximax,xmin:xstep:xmax,pk);
xlabel('xi');
ylabel('x');
title('Contour of the estimated joint density');

drawnow;

% store different plots in different files

if toprinter==1
    if k== -1,
        print -dps fig/figi.ps
    elseif k==0,
        print -dps fig/fig0.ps
    elseif k==1,
```

```
    print -dps fig/fig1.ps
elseif k==2,
    print -dps fig/fig2.ps
elseif k==3
    print -dps fig/fig3.ps
elseif k==4
    print -dps fig/fig4.ps
elseif k==5
    print -dps fig/fig5.ps
elseif k==6
    print -dps fig/fig6.ps
elseif k==7
    print -dps fig/fig7.ps
elseif k==8
    print -dps fig/fig8.ps
end
end
```