HÖGSKOLAN
TROLLHÄTTAN·UDDEVALLA
INSTITUTIONEN FÖR TEKNIK

EXAMENSARBETE

LQ-reglering av inverterad pendel

Micael Karlsson
H97Ei

2000-07-04

Högskolan Trollhättan-Uddevalla
Institutionen för Teknik
Box 957, 461 29 Trollhättan
Tel: 0520-47 50 00 Fax: 0520-47 50 99

EXAMENSARBETE

LQ-reglering av inverterad pendel

Sammanfattning

Med hjälp av MATLAB och tillhörande hårdvara har en LQ-regulator tagits fram som skall balansera en inverterad pendel i upprätt läge. Ett antal filer för att i en grafisk miljö samla mätdata för pendelsystemet har tagits fram för att kunna jämföra den matematiska modellen som levererats av företaget FEEDBACK med den mätdata som samlats in.
För att på ett enkelt sätt se hur LQ-reglering fungerar skapades en grafisk miljö. Med hjälp av steg- och pulssvar samt en animering där man kan studera hur vagnen och pendeln rör sig. Vid simulering är det möjligt att addera en störning till börvärdet.

Nyckelord: LQ, linjärkvadratisk reglering, reglerteknik, MATLAB, GUI, inverterad pendel

Utgivare: Högskolan Trollhättan/Uddevalla, Institutionen för Teknik
Box 957, 461 29 Trollhättan
Tel: 0520-47 50 00 Fax: 0520-47 50 99 E-post: teknik@htu.se

Författare: Micael Karlsson

Examinator: Anna-Karin Christiansson

Handledare: Anna-Karin Christiansson, HTU

Poäng: 10 **Nivå:** C

Huvudämne: Elektroteknik **Inriktning:** Informationssystem

Språk: Svenska **Nummer**: 2000:72 **Datum**: 2000-07-04

DISSERTATION

LQ-control of the inverted pendulum

Summary

The intention of this dissertation is the creation of a Linear Quadric Regulator (LQR) in MATLAB to control an inverted pendulum in an upright position. A number of files have been created in a graphical environment for collecting data from the plant. This data is used to verify the model given by the company that delivered the pendulum equipment.
To be familiarised with the LQ-control, a graphical interface has been created so the user can do experiments with different weight-factors.
It is possible to add noise to the desired value.

Keywords: LQ Control, MATLAB, GUI, Inverted pendulum

Publisher: University of Trollhättan/Uddevalla, Department of Technology
Box 957, S-461 29 Trollhättan, SWEDEN
Phone: + 46 520 47 50 00 Fax: + 46 520 47 50 99 E-mail: teknik@htu.se

Author: Micael Karlsson

Examiner: Anna-Karin Christiansson

Advisor: Anna-Karin Christiansson, HTU

Subject: Electrical Engineering, Information Systems

Language: Swedish **Number**: 2000:72 **Date**: July 4, 2000

Förord

Jag vill tacka min handledare Anna-Karin Christiansson för all den hjälp och stöttning jag fått under examensarbetets gång.

Innehållsförteckning

1 INLEDNING 6

1.1 Bakgrund 6

1.2 Syfte 6

1.3 Mål 6

1.4 Avgränsningar 6

1.5 Metod 7

1.6 Tillvägagångssätt 7

1.7 Källdiskusion 7

1.8 Litteratursökning 8

2 PENDEL-VAGNSYSTEMET 8

3 TILLSTÅNDSÅTERKOPPLING 10

3.1 Tillståndsrekonstruktion 12

4 LINJÄRKVADRATISK REGLERING 13

4.1 LQ - optimering i det tidsdiskreta fallet 14

4.2 LQ – optimering för tidskontinuerliga fall 15

5 VALIDERA DEN BEFINTLIG MODELLEN 16

6 DEN EGNA LQ-REGULATOR 23

6.1 Realisera regulatorn i MATLAB/SIMULINK 23

6.2 Simulering av modellen 24

7 REALTIDSKÄRNAN, RTK 28

8 MATLABS GRAFISKA GRÄNSSNITT, GUI 30

9 RESULTAT/SLUTSATSER 32

1 Inledning

1.1 Bakgrund

Denna rapport ingår som ett 10p examensarbete på akademisk C-nivå, detta för att uppnå kandidatexamen från Elektro-programmet med inriktning mot informationssytem på institutionen för Teknik vid HTU.

1.2 Syfte

Det övergripande målet med detta examensarbete har varit att fördjupa kunskaperna i reglerteknik samt fördjupade kunskaper i MATLAB.

1.3 Mål

Målen med examensarbetet har varit följande:

- Studera linjärkvadratisk reglering
- Studera en inverterad pendel som finns på institutionen
- Validera befintlig modell av pendeln genom ett antal tester
- Skapa en fungerande LQ-regulator till den inverterade pendeln
- Behärska MATLABs grafiska miljö och använda *Guide Control Panel,* ett hjälpmedel som finns tillgängligt i MATLAB, se vidare bilaga 4
- Dokumentera arbetet i form av en rapport.

1.4 Avgränsningar

Enbart den pendel och kringutrustning av märket FEEDBACK som finns på institutionen har använts. Då ett grafiskt användargränssnitt för insamling av mätdata testades var det uppenbart att den dator som använts tidigare var för långsam, dess data är:
Intel 486, 33 MHz
RAM-minne: 32 Mb
Grafikkort: 1 Mb minne
Med anledning av detta installerades mjuk- och hårdvara i en snabbare dator.

1.5 Metod

Efter genomgång av relevant litteratur som behandlar linjärkvadratisk optimering har detta ämne studerats ingående för att skaffa en god bas varpå hela examensarbetet har baserats. Den litteratur som studerats mest är B. Schmidtbauers bok ”Modellbaserade reglersystem” [1]. Dessutom har ett antal elektroniska källor [11-15] använts för att få en vidare bild av ämnena LQ och Modellbaserad reglering, kurslitteratur från tidigare kurser som är relevanta [2-5] samt diverse annan litteratur [6-10] som varit lämplig.

1.6 Tillvägagångssätt

Först av allt studerades den befintliga utrustningen och den dokumentation som finns tillgänglig från tillverkaren FEEDBACK. Då det visade sig att det fanns mjukvara för en nyare version av MATLAB (5.x) installerades denna tillsammans med MATLAB 5.3 på den befintliga datorn en äldre dator med Intels 486-processor. Då en del i målbeskrivningen var att använda MATLABs grafiska gränssnitt skapades ett sådant för att testa det öppna systemet då vagnen med dubbelpendeln användes som en kran/travers. Det visade sig då att denna dator även för personer med mycket stort tålamod tappade detsamma och blev ordentligt irriterade över den ”sega” datorn. Med anledning av detta installerades hård- och mjukvara i en ny dator av modernare snitt.

Den nya dator (beteckning: Regler 9, i Styr- och reglerlaborationssal: D204) som används är betydligt snabbare, dess data:
Intel Pentium III 550 MHz
RAM-minne: 128 Mb
Grafikkort: 8 Mb minne
Då denna dator var utrustad med Windows 98 uppstod ett antal problem som härrörde från operativsystemet. Då detta byttes ut och Windows 95b installerades försvann problemen. De programvaror som installerades var MATLAB 5.3 (R11.1) och programvara från tillverkaren för version 5.x av MATLAB.

1.7 Källdiskusion

Då materialet som anskaffats genom biblioteket [6-9] och via Internet [11-15] studerats, kunde det redan efter inte ens hälften av materialet konstateras att det finns en uppsjö olika modeller av den inverterade pendeln. Olikheterna kan med största sannolikhet tillskrivas de skiftande användningsområden modellerna tagits fram för. Det som till

en början vållade visst huvudbry var att för samma sak användes ett antal olika beteckningar. Men detta undanröjdes då författaren blev mer varm i kläderna och inte var i så stort behov av just beteckningar. Då kunskaperna till största delen hämtats från böcker och vad som kan betecknas som förhållandevis säkra källor på Internet, olika svenska och några utländska universitet, har den kritiska granskningen känts överflödig.

1.8 Litteratursökning

Som nämnts tidigare har institutionens bibliotek besökts ett flertal gånger för att låna böcker [6-9] i ämnet reglerteknik samt ett stort antal sökningar på Internet där det visade sig vara förvånansvärt enkelt att hitta tillförlitliga källor i ämnet LQ-reglering och modellbaserad reglering. Då detta tillsammans gav mer information än vad som behövdes fanns det inget egentligt behov av att söka igenom det stora utbud av databaser som stod till förfogande.

2 Pendel-vagnsystemet

Det reglerobjekt som använts är av märket FEEDBACK ”Digital Pendulum Mechanical Unit 33-200” med tillhörande programvara från samma företag. Det är även deras realtidskärna[1] som används. Systemet består av en DC-motor med ett drivhjul kopplat till motoraxeln som driver vagnen fram och tillbaka med hjälp av ett drivband. Till vagnen är det kopplat två pendelarmar, en på var sida av vagnen. Dessa är dock sammankopplade med en horisontell metallstav varför de kan betecknas som en enda pendel. Det finns även två inkrementella givare, en för vagnens läge och en för pendelarmarnas vinkeln. Vinkelgivaren har enligt tillverkaren en upplösning på:

$$\Delta j = \frac{2p}{2048} = 0{,}00307\,[\text{rad}]$$

En enkel bild av den fysiska utrustningen ges i figur 1.

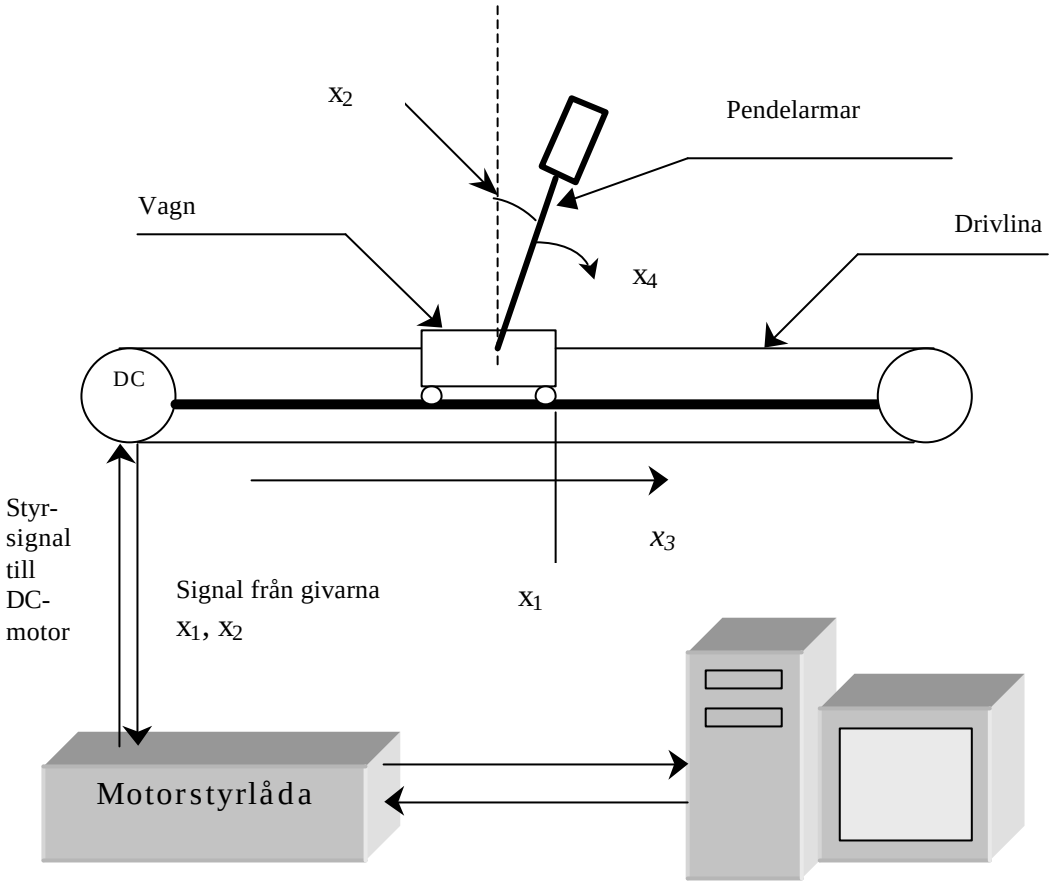
Denna givare har den egenskapen att då pendelarmarna roterar flera varv ökas vinkelvärdet. Detta ger problem vid implementering av den egna regulatorn eftersom vinkeln för att uppfylla de kriterier som ställts skall vara nära noll.

[1] Se bilaga 5 hur denna anropas i MATLAB

Lyckligtvis går det troligtvis att gå förbi detta problem genom att i sina beräkningar använda följande rad MATLAB-kod:

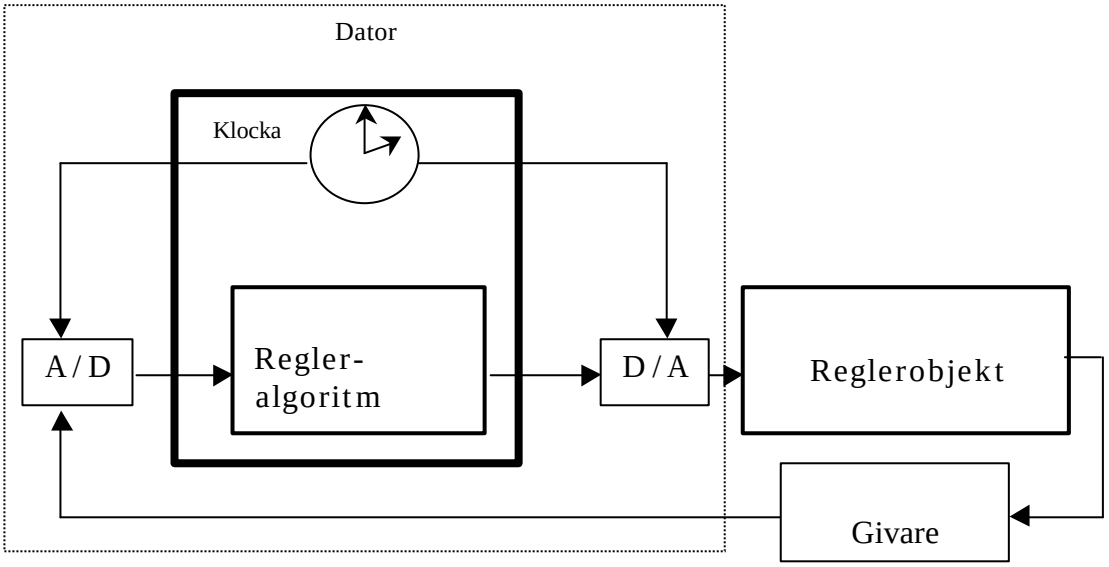
```
alfa = pi*((alfax/pi)-floor(alfax/pi))
```

där `alfax` är vinkelvärdet från givaren och `alfa` det vinkelvärde som används för beräkningarna av styrsignalen.



Figur 1. *Bild av hur hårdvaran är hopkopplad*

Den fysiska representation i figur 1 av systemet kan byggas upp med hjälp av ett antal block som på ett logiskt sätt beskriver hur den datorbaserade regleringen går till, se figur .



Figur 2. *Blockschemamässig beskrivning av datorbaserad reglering.*

3 Tillståndsåterkoppling

Om alla tillstånd, det vill säga läge, hastighet, vinkel och vinkelhastighet för reglerobjektet[2] finns att tillgå som mätsignaler kan dessa återkopplas och multipliceras med en viktfaktor för varje tillstånd. Det är tanken att dessa viktfaktorer väljs så att polerna för det återkopplade systemet får de önskade värdena, det är detta som kallas polplacering. Valet av dessa poler för det slutna systemet blir då en kompromiss mellan snabbhet/noggrannhet och styrsignalnivåer/robusthet. Denna metod fungerar bäst på slutna system av SISO-typ (Single Input - Single Output). Då flera styrsignaler finns, MIMO (Multiple Input - Multiple Output) hänvisas i litteratur till linjärkvadratisk optimering [1].

Som synes i figur 3 skall reglerobjektet finnas angivet på tillståndsform (eller A,B,C,D-form). Det som först måste åstadkommas för att arbeta med denna form av reglering är att om det reglerobjektet består av en ickelinjär matematisk modell skall denna linjäriseras.

[2] Här avses endast den inverterade pendeln

Det block i figur 3 som betecknas *Kr* är börvärdesfaktor som beräknas så att lågfrekvensförstärkningen från *R* till Y är 1. För att uppnå detta sätts K_r till:

$$K_r = \left[C(-A^T)^{-1}B\right]^{-1} \tag{3.1}$$

i det kontinuerliga fallet och

$$K_r = \left[C\left(I - A_d{}^T\right)^{-1}B_d\right]^{-1} \tag{3.2}$$

i det tidsdiskreta fallet. *I* är enhetsmatris med samma dimension som A_d, den diskreta motsvarigheten till *A*. B_d motsvaras i det kontinuerliga fallet av *B*. Diskretiseringen av tillståndsmodellen görs enkelt i MATLAB med kommandot: `c2dm`. De inparametrar som ges är den kontinuerliga tillståndsmodellens fyra matriser, samplingstiden, och vilken diskretiseringsmetod som skall användas.

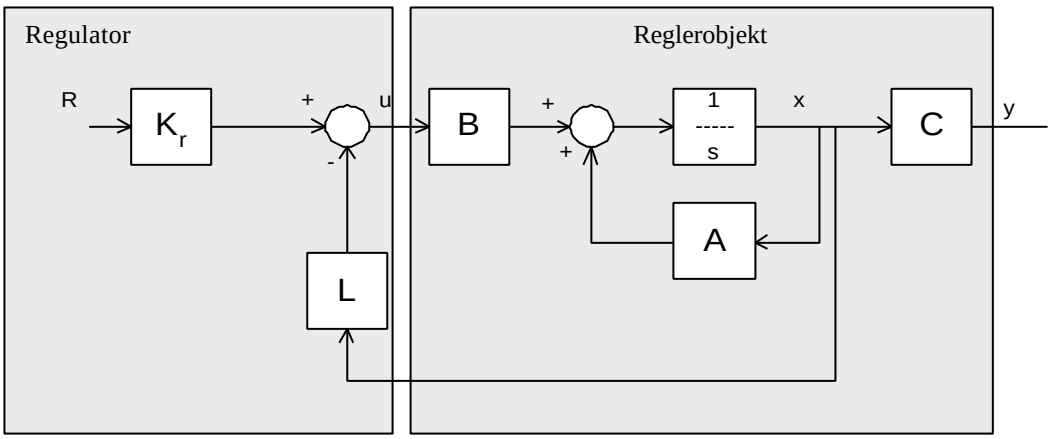
Styrlagen *L* i figur 3 beräknas enligt följande:

$$\begin{aligned} L &= [l_1, l_2, ..., l_n] \\ A_d' &= A_d - B_d L \end{aligned} \tag{3.3}$$

det karakteristiska polynomet ansätts enligt följande:

$$P(z) = \det(zI - A_d') \tag{3.4}$$

ur detta polynom fås ett antal ekvationer med *z* och $l_1, l_2 .. l_n$ som okända. Variabeln *z* ges de värden som man önskar att polerna för det slutna systemet skall få och löser ut $l_1, l_2 .. l_n$ som alltså bildar styrlag *L*. Jämför med kapitlen 4.1 och 4.2 som behandlar LQ-reglering där avvikelser straffas i stället för att som här placera poler.

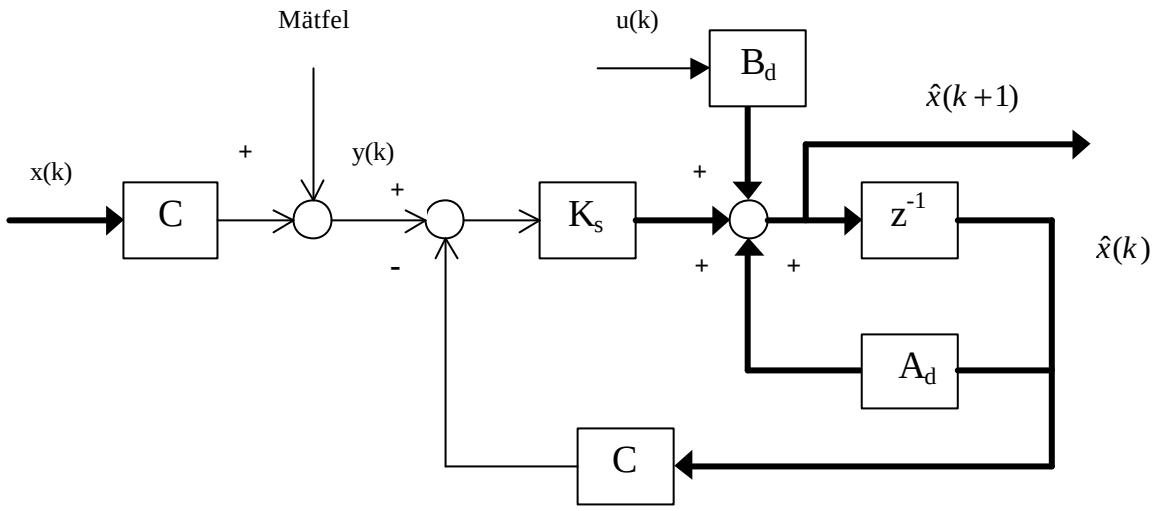


Figur 3. *Blockschema för tillståndsmodell med tillstånds-återkoppling via styrlagen L, där u = K_r*R-Lx*

Då denna typ av regulatorer ofta realiseras med hjälp av datorer är de tidsdiskreta. Det problem som kan uppstå är om samplingstiden är kort eller om beräkningarna tar så pass mycket tid att ett helt samplingsintervall går åt bör man använda det som kallas ”långsam dator” [1]. Det har alltså inget med den använda datorns prestanda att göra.

3.1 Tillståndsrekonstruktion

Då det inte alltid är möjligt att mäta alla tillstånd i reglerobjektet återskapas de med hjälp av en modell av reglerobjektet. Det är alltså av största vikt att ha en bra modell för att regulatorn skall fungera på ett tillfredsställande sätt. De estimerade tillstånden betecknas $\hat{x}(k)$ (x-hatt) för att skilja dem från de verkliga tillstånden *x*. I figur 4 beskrivs det som betecknas ettstegs-prediktion eller ”långsam dator”[1].



Figur 4. *Tidsdiskret observatör , ettstegsprediktion*

Man inför beteckningen $\hat{x}(k|k-1)$ för den tillståndsuppskattning som görs vid samplingstillfälle *k* baserat på mätningen vid samplingstillfälle *(k-1).* Man baserar alltså tillstånds-uppskattningen på mätningar fram till tidpunkten *(k-1)* för att få en korrigering av tillståndsskattningen:

$$\hat{x}(k|k) = A_d\hat{x}(k|k-1) + K_s\left[y(k) - C\hat{x}(k|k-1)\right] \qquad (3.5)$$

med hjälp av denna formel sker alltså prediktering ett steg framåt i tiden, detta baseras på $\hat{x}(k|k)$. Inför beteckningen

$$\hat{x} \equiv \hat{x}(k|k-1) \tag{3.6}$$

Med dessa omskrivningar fås:

$$\hat{x}(k+1) = (A_d - K_s C)\hat{x}(k) + Gu(k) + K_s y(k) \tag{3.7}$$

som slutresultat.

K_s beräknas enligt följande [1]:

$$K_s = A_d^{-1} K \tag{3.8}$$

4 Linjärkvadratisk reglering

På 1960-talet utvecklades det som nu betecknas LQ-reglering. Det utvecklades med stora forskningsanslag i USA och dåvarande Sovjetunionen för rymdrelaterade reglerproblem. Dessa problem bestod bland annat av manövrering av raketer för att minimera bränsleåtgång. Särskilt inom rymdfarten var tekniker lyckosamma med att implementera linjärkavdratisk Gaussisk reglering, LQG-reglering. Då reglertekniker inom andra branscher försökte använda LQG gick det tyvärr inte lika bra eftersom det oftast inte fanns tillräckligt bra modeller av reglerobjekten. Det fanns dessutom ingen bra modell av det vita brus som fanns närvarande. Med anledning av detta var det inte alltid dessa regulatorer var tillräckligt robusta.[9]

Linjärkvadratisk(LQ) optimering kan sägas vara den optimala regulatorn för ett reglerobjekt, stabilt eller instabilt. Det är dock nödvändigt att ha en ”bra” matematisk modell av reglerobjektet om det inte finns möjlighet att direkt mäta alla tillstånd. Tanken med LQ är att straffa avvikelser från det önskade värden. Dessa straff sätts utifrån vilka tillstånd som är mest kritiska för funktionaliteten. Som i fallet med den inverterade pendeln där vagnens läge inte alltid kan anses vara det viktigaste utan pendelns avvikelse från dess upprätta läge. Då är det lämpligt att i förhållande till straffet för vagnens läge sätta ett större straff för vinkelavvikelse. Straffen bör dock sättas inom vissa gränser så att styrsignalen/styrsignalerna finns inom ett rimligt område. Det kan annars bli så att styrsignalen antar allt för stora värden vilket leder till försämrad reglering då en sådan signal inte

kan överföras till den utrustning som direkt påverkar reglerobjektet som t.ex. en motor eller liknande.

LQ-reglering kan beskrivas på flera sätt. Ett av dem är nedanstående [11]

”I t.ex. en autopilot till en segelbåt vill man gärna ligga bra i kurs utan att dra ur batterierna helt. Då kan ett stort straff på kursavvikelse och ett litet på styrenergin göra att man får en utmärkt kurs till kostnad av att batteriet tar slut på en kort tid. Om man då gör omvänt, stort straff på styrenergi och lågt på kursavvikelse är det troligt att man hamnar på land med fulladdade batterier. Målet är ju att man skall hålla en acceptabel kurs utan att dra ur batterierna, detta uppnås med att man balanserar straff för avvikelse i kurs och använd styrenergi så att man hamnar där man vill”.

4.1 LQ - optimering i det tidsdiskreta fallet

Den diskreta tillståndsmodellen för ett system av ordning *n* ges av följande modell:

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{4.1}$$

Detta system skall styras med *u(k)* i *N* antal steg från *k=0* till *k=N-1*, med givet begynnelsetillstånd *x(0)* så att kostnadsfunktionen *J* minimeras:

$$J = \sum_{k=0}^{N-1} \left[x^T(k) Q_1 x(k) + u^T(k) Q_2 u(k) \right] + x^T(N) M x(N) = \min! \tag{4.2}$$

I (4.2) är Q_1 straffen för tillstånden och Q_2 för styrsignalen / styrsignaler och *M* är straffet för det kvarstående fel. Först optimeras kostnaden i sista steget för att sedan arbeta sig bakåt till första steget, detta kallas *dynamisk programmering*. Styrsignalen för $u(N-1) \equiv -L(N-1)x(N-1)$, och kostnadsfunktionen

$$J_{N-1} \equiv x^T(N-1) S(N-1) x(N-1) \tag{4.3}$$

För beräkning av styrlagen *L(k)* och kostnadsfunktionen *S(k)*

$$L(k) = \left[B_d{}^T S(k+1)B_d + Q_2\right]^{-1} B_d{}^T S(k+1)A_d \tag{4.4}$$

$$\begin{aligned} S(k) =& A_d{}^T S(k+1)A_d - \\ &- A_d{}^T S(k+1)B_d \left[B_d{}^T S(k+1)B_d + Q_2\right]^{-1} B_d{}^T S(k+1)A_d + Q_1 \end{aligned} \tag{4.5}$$

Minimivärdet av kostnadsfunktionen blir:

$$J_{\min} = x^T(0)S(0)x(0) \tag{4.6}$$

Om styrtiden är lång fås det stationära värdet på *S* som lösningen till den stationära tidsdiskreta Riccati-ekvationen:

$$S = A_d{}^T SA_d - A_d{}^T SB_d \left[G^T SB_d + Q_2\right]^{-1} B_d{}^T SA_d + Q_1 \tag{4.7}$$

I MATLAB finns rutiner för att beräkna lösningen till Riccati-ekvationen. Det kommando som används i det diskreta fallet kallas: `dare`. Det finns naturligtvis motsvarande kommando för det kontinuerliga fallet: `are`. Formlerna (4.1)-(4.7) är hämtade ur ”Modellbaserade reglersystem” av B. Schmidtbauer.

4.2 LQ – optimering för tidskontinuerliga fall

Den tidskontinuerliga tillståndsmodellen beskrivs på nedanstående sätt:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{4.8}$$

Motsvarande optimeringskriterium

$$J = \int_0^T \left[x^T(t)Q_1x(t) + u^T(t)Q_2u(t)\right]dt + x^T(T)Mx(T) \tag{4.9}$$

skall minimeras, där Q_1 (tillståndsstraff) och Q_2 (styrsignalstraff) är kostnadsmatriserna samt M är straffet för tillståndsavvikelse vid sluttidpunkten (*T*).

Styrlagen i det tidskontinuerliga fallet är som följer:

$$L(t) = Q_2^{-1} B^T S(t) \tag{4.10}$$

Helt analogt med det tidsdiskreta fallet ges minimivärdet av kostnadsfunktionen av:

$$J_{\min} = x^T(0) S(0) x(0) \tag{4.11}$$

Om det tidskontinuerliga systemet $\dot{x} = Ax + Bu$ är möjligt att stabilisera kommer Riccati-ekvationen

$$-\dot{S} = SA + A^T S - SBQ_2^{-1} B^T S + Q_1 \tag{4.12}$$

att konvergera till en stationär matris S enligt:

$$SA + A^T S - SBQ_2^{-1} B^T S + Q_1 = 0 \tag{4.13}$$

För att erhålla den konstanta styrlagen sätts S in i (4.10). Då dessa beräkningar inte blir helt enkla vid större matriser än 2x2 är det till stor hjälp att det i MATLABs Control System Toolbox (CSTB) finns kommandon för att beräkna den konstanta styrlagen och stationära Riccati-ekvationen. Formlerna (4.8)-(4.13) är hämtade ur ”Modellbaserade reglersystem” av B. Schmidtbauer.

5 Validera den befintlig modellen

De fysikaliska samband som ligger till grund för modellen är hämtade ur den dokumentation som levereras med utrustningen. Denna modell kan ses som den erhållits från domänexpert varför modellen accepteras som den är. I dessa ekvationer används de storheter som redovisas i tabell 1.

Tabell 1. *Parametrar för pendel-vagn modellen*

Parameternamn	**FEEDBACKs värde**	**Identifierat/mätt värde**
Banans längd	±0,5 m	±0,5 m
Gravitation, g	9,81 m/s^2	9,81 m/s^2
Avstånd mellan masscentrum och rotationsaxeln, l	0,017 m	0,01679 m
Vagnens massa, m_c	1,12 kg	1,06 kg
Pendelarmens massa, m_p	0,11 kg	0,19 kg
Systemets tröghet, J	0,0136 kgm^2	0,01556 kgm^2
Friktionskoefficient i pendelarmarnas rotationspunkt, f_p	Negligerbar	0,0002 Ns/m
Vagnens friktionskoefficient, f_c	0,05 Ns/m	0,05 Ns/m
Banans motriktade kraft, V	---	---
Rörelsefriktionen för vagnen som beror av vagnens hastighet, T_c	---	---

Värdena i kolumnen ”FEEDBACKs värde” är hämtade från företagets dokumentation. I kolumnen ”Identifierat/mätt värde” har värdena för vikterna på vagn och pendelarmen vägts med hjälp av våg. De andra har identifierats med FEEDBACKs identifieringsprogram som redovisas i bilaga 1.

De tillstånd som valts är följande, se även figur 3:

x_1 = Vagnens position [m]
x_2 = Pendelns vinkel [rad]
x_3 = Vagnens hastighet [m/s]
x_4 = Pendelns vinkelhastighet [rad/s]

För definition av storheter se tabell 1.

Ekvationerna (5.1) är vad FEEDBACK beteckner ”equations of motion”

$$\frac{d^2}{dt^2}\left(m_c + m_p\right)\left(x_1 - l\sin x_2\right) = F - T_c$$
$$\frac{d^2}{dt^2}\left(m_c + m_p\right)\left(l\cos x_2\right) = V - \left(m_c + m_p\right)g \qquad (5.1)$$
$$Jx_2 = \left(F - T_c\right)l\cos x_2 + Vl\sin x_2 - f_p x_4$$

Ur ekvationerna (5.1) som hämtats från FEEDBACKs dokumentation kan följande modell för pendelsystemet tas fram.

$$\dot{x}_1 = x_3 = f_1(x,u)$$
$$\dot{x}_2 = x_4 = f_2(x,u)$$
$$\dot{x}_3 = \frac{a\left(u - Tc - mx_4{}^2\sin x_2\right) + l\cos x_2\left(mg\sin x_2 - f_p x_4\right)}{J + ml\sin^2 x_2} = f_3(x,u) \qquad (5.2)$$
$$\dot{x}_4 = \frac{l\cos x_2\left(u - Tc - mx_4{}^2\sin x_2\right) + mg\sin x_2 - f_p x_4}{J + ml\sin^2 x_2} = f_4(x,u)$$

där:

$$a = l^2 + \frac{J}{m_c + m_p}$$
$$m = l\left(m_c + m_p\right) \qquad (5.3)$$
$$T_c = f_c x_3$$
$$u = insignal$$

Förutsättningar för linearisering, $x_2 = \alpha$

$\alpha \approx 0$: $\sin\alpha \approx \alpha$, $\cos\alpha \approx 1$, det vill säga pendelarmarna är i upprätt läge.

Linjärisering kring den stationära punkten *SP* blir:

$$\Delta\dot{x} \approx \left.\frac{d\,f(x,u)}{d\,x_1}\right|_{SP}\cdot\Delta x_1 + \left.\frac{d\,f(x,u)}{d\,x_2}\right|_{SP}\cdot\Delta x_2 + \left.\frac{d\,f(x,u)}{d\,x_3}\right|_{SP}\cdot\Delta x_3 + \qquad (5.4)$$
$$+ \left.\frac{d\,f(x,u)}{d\,x_4}\right|_{SP}\cdot\Delta x_4 + \left.\frac{d\,f(x,u)}{d\,u}\right|_{SP}\cdot\Delta u$$

$$
\begin{aligned}
\Delta\dot{x}_1 &\approx \Delta x_3 \\
\Delta\dot{x}_2 &\approx \Delta x_4 \\
\Delta\dot{x}_3 &\approx 0\Delta x_1 + \\
&\left. + \frac{\left(J + ml\sin^2 x_2\right)\left(-amx_4{}^2\cos x_2 + l\left(mg\left(s\cos^2 x_2 - l\right) + fpx_4{}^2\sin^2 x_2\right)\right)}{\left(J + ml\sin^2 x_2\right)^2}\right|_{SP} \cdot \Delta x_1 + \\
&+ 0\cdot\Delta x_3 - \left.\frac{2ax_4\sin x_2 m - lf_p\cos x_2}{J + ml\sin^2 x_2}\right|_{SP} \cdot \Delta x_4 + \left.\frac{a}{J + ml\sin^2 x_2}\right|_{SP} \cdot \Delta u \\
\Delta\dot{x}_4 &\approx 0\cdot\Delta x_1 + \left.\frac{\left(J + ml\sin^2 x_2\right)\left(-l\left(2\cos^2 x_2 - 1\right)x_4{}^2 u + mg\cos x_2\right)}{\left(J + ml\sin^2 x_2\right)^2}\right|_{SP} \cdot \Delta x_2 + \\
&+ 0\cdot\Delta x_3 - \left.\frac{lmx_4\sin(2x_2) + f_p}{J + ml\sin^2 x_2}\right|_{SP} \cdot \Delta x_4 + \left.\frac{l\cos x_2}{J + ml\sin^2 x_2}\right|_{SP} \cdot \Delta u
\end{aligned}
\qquad (5.5)
$$

Det är endast intressant att använda den linjäriserade modellen (5.5) då vinkel är nära noll, det vill säga då pendelarmarna är i upprätt läge. I (5.6) har α ersatt tillståndet x_2.

$$
\begin{aligned}
\Delta\dot{x}_1 &\approx \Delta x_3 \\
\Delta\dot{x}_2 &\approx \Delta x_4 \\
\Delta\dot{x}_3 &\approx \frac{\left(J + ml(a)^2\right)\left(l\left(mg\left(2 - l\right)\right)\right)}{\left(J + lm(a)^2\right)^2}\cdot\Delta x_2 + \frac{lf_p}{J + lm(a)^2}\cdot\Delta x_4 + \\
&+ \frac{a}{J + lm(a)^2}\cdot\Delta u \\
\Delta\dot{x}_4 &\approx \frac{\left(J + lm(a)^2\right)mg}{\left(J + lm(a)^2\right)^2}\cdot\Delta x_2 - \frac{f_p}{J + lm(a)^2}\cdot\Delta x_4 + \\
&+ \frac{l}{J + lm(a)^2}\cdot\Delta u
\end{aligned}
\qquad (5.6)
$$

Ur denna linjäriserade tillståndsmodell kan A,B,C,D-matriserna lätt plockas ut då den allmänna tillståndsmodellen är på följande form:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{\left(J + ml(a)^2\right)(l(mg(2-l)))}{\left(J + lm(a)^2\right)^2} & 0 & \dfrac{lf_p}{J + lm(a)^2} \\ 0 & \dfrac{\left(J + lm(a)^2\right)mg}{\left(J + lm(a)^2\right)^2} & 0 & -\dfrac{f_p}{J + lm(a)^2} \end{bmatrix} \qquad (5.7)$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \dfrac{a}{J + lm(a)^2} \\ \dfrac{l}{J + lm(a)^2} \end{bmatrix}; \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}; \quad D = [0]$$

$$\dot{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0{,}5953 & 0 & 0{,}0125 \\ 0 & 17{,}6580 & 0 & 0{,}7353 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0{,}7157 \\ 1{,}25 \end{bmatrix} u \qquad (5.8)$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x$$

med värden från tabell 1, kolumn ”identifierat/mätt värde” har dessa matrisers element beräknats för $\alpha = 0$.

I dokumentationen från tillverkaren FEEDBACK finns en SIMULINK-fil med som gör det möjligt att med hjälp av MATLAB-kommandot `linmod` lätt skaffa sig en linjär tillståndsmodell av den olinjära SIMULINK-modellen model4lq.mdl, se bilaga 2. I MATLABs kommandofönster ges följande kommando:

```
[Au,Bu,Cu,Du]=linmod('model4lq',[0,0,0,0],0);
```

fås följande värden på Au, Bu, Cu, Du.

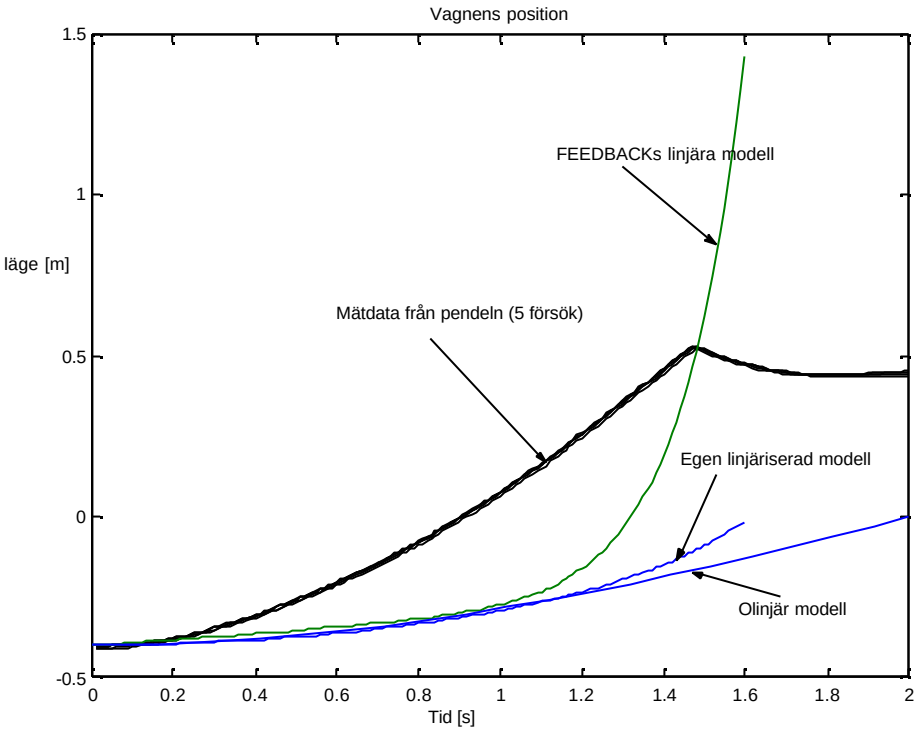
$$Au = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -34{,}9727 & -0{,}0001 \\ 0 & 43{,}43113 & -145{,}7195 & -0{,}0054 \end{bmatrix}$$

$$Bu = \begin{bmatrix} 0 \\ 0 \\ 9{,}1803 \\ 38{,}2514 \end{bmatrix}; \qquad Cu = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.9)$$

$$Du = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Variablerna Au,Bu,Cu,Du är för den tidskontinuerliga linjära tillståndsmodellen av SIMULINK-fil, model4lq.mdl, denna redovisas i bilaga 2.

En jämförelse mellan modellerna (5.2), (5.8), (5.9) och mätdata från pendeln redovisas i figur 5. Samma steg har givits som insignal.



Figur 5. *Jämförelse av olika modellers resultat i jämförelse med mätdata.*

I figur 5 är det lätt att se att de två linjära modellerna stämmer väl överens men den olinjära modellen under c:a 1 s. Däremot är de efter c:a 0.2 s som modellerna skiljer sig från mätdatan i allt större grad ju längre tiden går. I intervallet 1,4 – 1,6 s slår vagnen emot ena ändläget och efter det bör därför inga jämförelser göras. Det är dock med i figuren för att ge en uppfattning om hur modellerna ”uppför sig”.

Anledningarna till att de olika modellerna skiljer sig åt kan bero av en av nedanstående eller en kombination av några av dem:

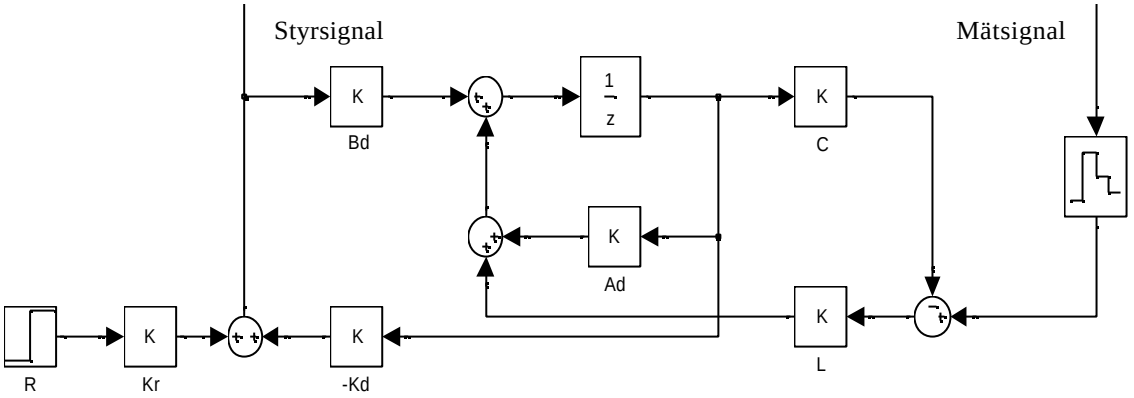
- Vagnen och pendelarmarnas vikt
- Friktion mellan vagnens hjul och banan där vagnen rullar
- Hur hårt drivbandet är spänt

Den sista punkten har under examensarbetets gång visat sig vara av stor vikt.

Figur 5 har skapats med hjälp av m-filerna jfr_egen_fbk.m och validera2.m för källkod se bilaga 2.

6 Den egna LQ-regulator

Den LQ-regulator som används har en observatör med tillståndsrekonstruktion och ettstegsprediktion för en så kallad ”långsam dator”. Detta beror på att samplingstiden är så pass kort, 0,01 s. Det tar den tiden att beräkna styrsignalen för nästa samplingsperiod (*k+1*). I SIMULINK kan denna regulator enkelt realiseras, se figur 6.



Figur 6. *SIMULINK- schema för LQ-regulator med ettstegsprediktion*

6.1 Realisera regulatorn i MATLAB/SIMULINK

Det finns två sätt att i SIMULINK realisera regulatorn som avbildas i figur 6, dels det som redovisas i den figuren dels kan hela regulatorn realiseras som en diskret tillståndsmodell med hjälp av blocket *”Discrete State Space”* där regulatorns A,B,C,D-matriser anges i blockets ”Block Parameters”. Se figur 7.
För att beräkna regulatorns tillståndsmodell bör följande steg tas:

1. Ansätt matrisen Q_1 och Q_2 det vill säga vikt-/straff-matriserna för tillstånden resp. styrsignalamplitud.
2. Uppskatta process (`Qn`)- och mätbrus (`Rn`)
3. För att använda kommandot `kalmd` måste tillståndsmodellen vara ett ss-objekt vilket görs med följande rad: `sys = ss(A,B,C,D)`
4. Beräkna tidsdiskret kalmanestimator[3] med hjälp av `kalmd. [Kest,L,P,M,Z] = kalmd(sys,Qn,Rn,Ts)` Detta kommando realiserar estimatorn utifrån den

[3] Samma som tillståndrekonstruktion, se kapitel 3.1

kontinuerliga processmodellen. Därefter beräknas tillståndsåterkopplingen med hjälp av `lqrd`.
`[Kd,S,e] = lqrd(A,B,Q1,Q2,Ts).`
Detta kommando realiserar en diskret tillståndsåterkoppling.

5. Därefter diskretiseras modellen med kommandot `c2dm` som ger en diskret tillståndsmodell av den kontinuerliga. `[Ad,Bd,Cd,Dd]= c2dm(A,B,C,D,Ts,'zoh')`, där `Ts` är samplingstid och `'zoh'` står för "`zero order hold`", dvs. en nollte ordningens hållkrets används.
6. Därefter beräknas de fyra tillståndsmatriserna enligt följande: `Areg = Adm - L*Cd - Bdm*Kd;`
   `Breg = L;`
   `Creg = -Kd;`
   `Dreg =[0 0];`

Då det visat sig vara ett sisyfosarbete att få en fungerande överföring av data från och till pendeln har detta övergivits efter flera timmars slit. För att testa regulatorn har detta gjorts mot den modell som levereras med dokumentationen. Denna regulator har tagits fram utifrån denna modell enligt ovanstående tillvägagångssätt. M-filen redovisas i bilaga 2 med filnamn: lqr_test.m.
De olika försöken som gjordes för att beräkna rätt styrsignal utifrån inhämtad data var följande:

1. Beräkna styrsignal i SIMULINK och överföra styrsignalamplituden till arbetsarean i MATLAB för att vidarebefordra den till pendeln
2. Med hjälp av kommandot `lsim` och föregående sampels tillstånd dvs. *(k-1)* beräkna styrsignalen och överföra denna.
3. Beräkna styrsignalen "för hand" i MATLAB.

Då det i bästa fall blev en reaktion från pendeln som bevisade att åtminstone något överförts var denna signal helt felaktig. Pendel och vagn brakade in i ena ändläget med hög fart. Det troligaste är att det inte är i beräkningarna av styrsignalen som problemet ligger utan just i kommunikationen mellan MATLAB och pendeln samt att läsa in rätt värden vid rätt tidpunkt. I samråd med handledaren avbröts de fruktlösa försöken med att överföra rätt mätdata till MATLAB och styrsignal till pendeln. I stället beslöts att enbart simulera med en mer verklighetstrogen SIMULINK-fil.
Regulatormodellen testades därigenom då brus adderades till både mät- och strysignal, se figur 7.

6.2 Simulering av modellen

I alla försöken i detta kapitel har regulatorn designats utifrån den ickelinjära modell som bifogats i dokumentationen från

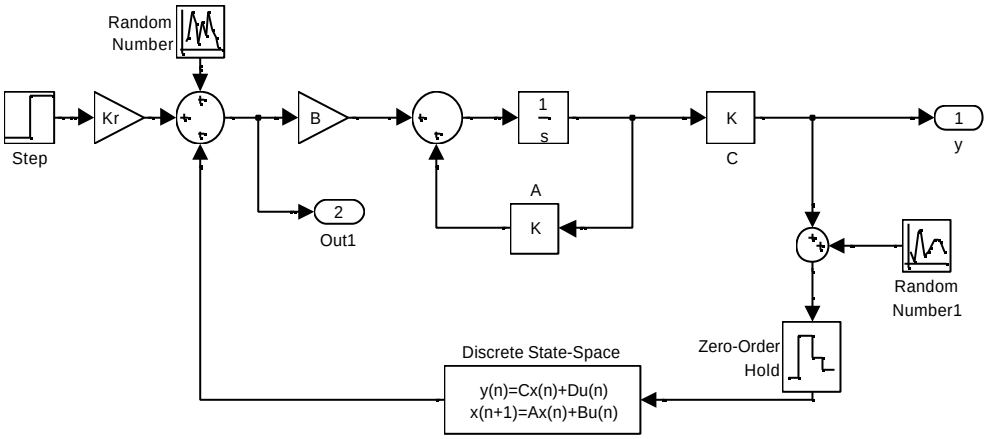
tillverkaren, SIMULINK-fil model4lq.mdl, för blockscheman se bilaga 2. Denna kan dock lätt linjäriseras med kommandot `linmod` då man anger kring vilken punkt modellen skall linjäriseras, i dessa fall med alla begynnelsevärden satta till noll(0). För att om möjligt efterlikna verkligheten har i SIMULINK-filen lagts in två block för att simulera brus i både styr- och mätsignal. För blocket *”Random number: Styrsignalbrus”* har följande data angetts:

Mean: 0
Variance: 0.00001
Initial seed:112233
Sample Time: 0.01 [s]

För ”Random number: Styrsignalbrus”

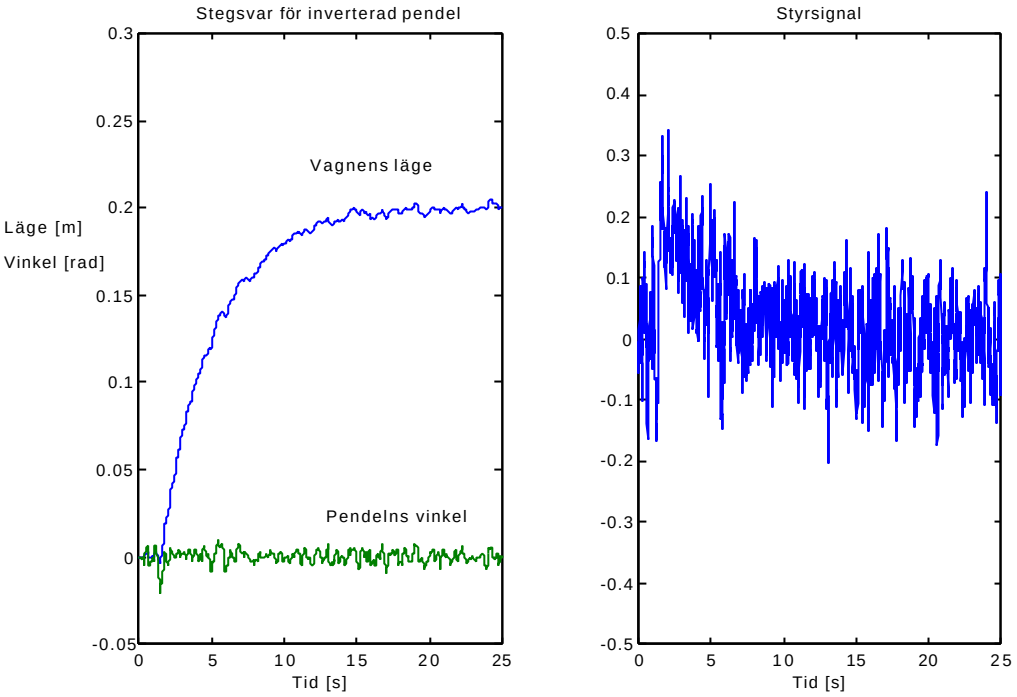
Mean: 0
Variance: 0.00001
Initial seed:112233
Sample Time: 0.01 [s]

I figur 7 redovisas den SIMULINK-fil som använts vid alla simuleringar i denna del av rapporten.

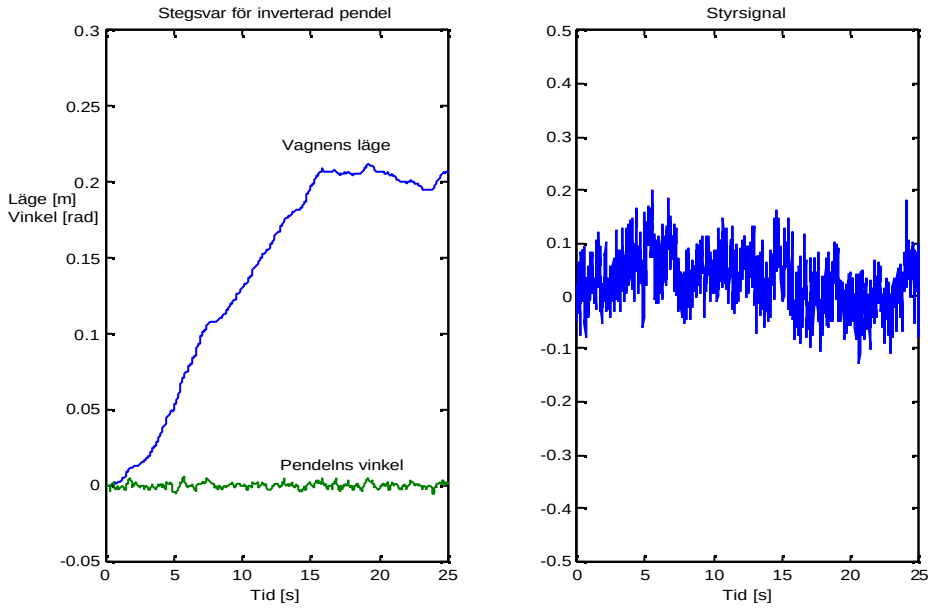


Figur 7. *SIMULINK-schema som använts vid simulering av LQ-reglering av inverterad pendel.*

Med straffet 1000 för vagnens läge och 1 för pendelns avvikelse från upprätt läge fås kurvorna som redovisas i figur 8. Med dessa straff prioriteras alltså vagnens läge vilket syns tydligt då kurvorna för vagnens läge jämförs i figurerna 8 och 9. Det samma gäller naturligtvis för pendelns vinkel.

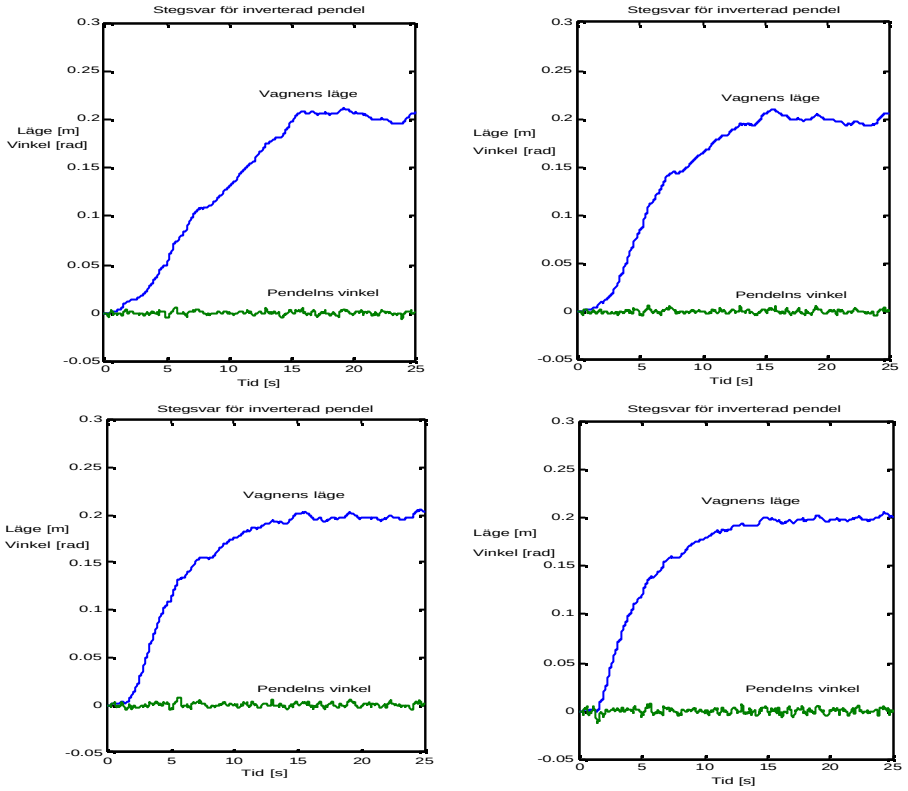


Figur 8. *Utsignal från pendeln (till vänster) och styrsignalen (till höger).*



Figur 9. *Utsignal från pendeln (till vänster) och styrsignalen (till höger).*

I figur 9 har straffen för vagnens läge satts till 1 och för pendelns avvikelse satts till 1000. Detta kan tyckas vara en dramatisk skillnad men den gör det enklare att observera skillnaden.

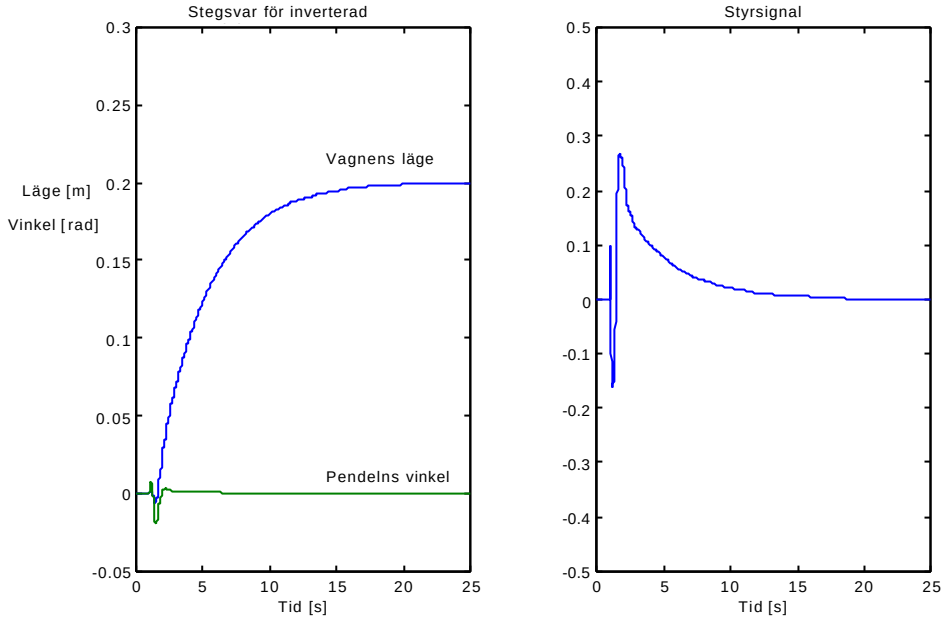


Figur 10. *Jämförelse av fyra olika stegsvar från den LQ-reglerade pendel-modellen.*

I figur 10 redovisas fyra olika straffkombinationer:

- Överst till vänster: vagnens straff: 1, pendeln: 1000
- Överst till höger: vagnens straff: 10, pendeln: 1000
- Nederst till vänster: vagnens straff: 100, pendeln:1000
- Nederst till vänster: vagnens straff: 1000, pendeln:1000

Om allt brus elimineras fås ett ”fint” uppträdande, se figur 11. Figur 11 är med för att illustrera hur reglerobjektet uppträder i en störningsfri miljö.

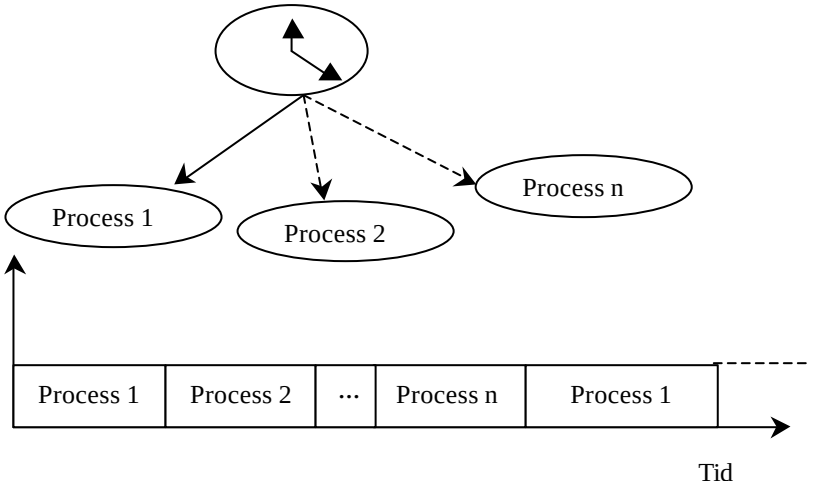


Figur 11. *Utsignal(till vänster) och styrsignal (till höger) då styrsignal- och mätbrus eliminerats, straff för vagn: 1000 och straff för vinkel: 1.*

7 RealTidsKärnan, RTK

Då RTK diskuteras används begreppet process som en beskrivning av olika delar av mjukvaran och inte som i reglerteknik är brukligt ett reglerobjekt.

Då de olika processerna, beräkning av styrsignal och avläsning av mätdata, sker i var sin process, behövs en realtidskärna och det är den som behandlas i detta kapitel. Med hjälp av figur 12 kan man på ett enkelt sätt få en uppfattning om hur en realtidskärna behandlar processer.

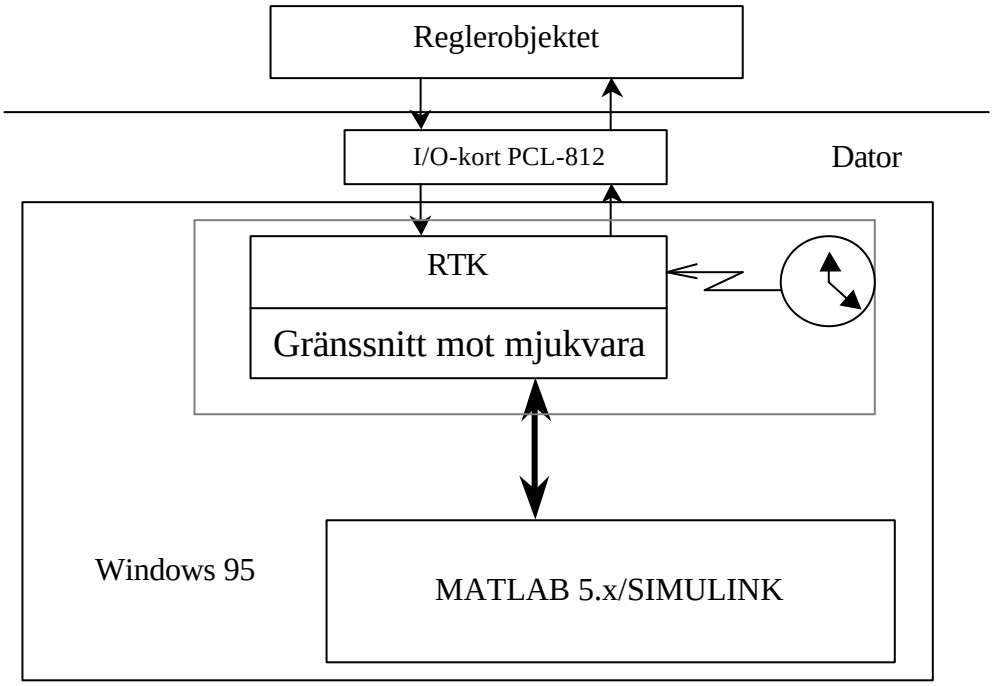


Figur 12. *Förenklad beskrivning av realtidsfunktion*

För att lösa kommunikationen, inhämtande av data och överföra styrsignal har företaget som levererat pendeln inkluderat en realtidskärna som anropas från MATLAB med kommandot `pd_call`, se vidare bilaga 5 för mer utförlig information om hur kommandot används. Exempel på syntax för kommandot `pd_call`:

```
hist = pd_call('GetHistory');
```

detta kommando ger den data som finns i bufferten, dock mest mätvärden vid 1024 mättillfällen. Realtidskärnan måste användas då operativsystemet som används är Windows 95b inte har funktioner för realtid utan detta måste ske via ett tredjepartsprogram, i detta fall från företaget FEEDBACK. I figur 13 ges en dock inte fullständig bild av en mer mjukvarumässig funktion för realtidskärnan som använts. Det är alltså denna del av examensarbetet som inte kunde lösas till fullo.



Figur 13. *Beskrivning av kommunikationen mellan ”reglerobjekt” och mjukvara*

8 MATLABs grafiska gränssnitt, GUI

I MATLAB finns dels ett antal verktyg samlade i *”Guide Control Panel”* för att utveckla ett grafiskt gränssnitt för användare. Denna ”verktygslåda” innehåller i stort sett allt som behövs för att utveckla ett fungerande gränssnitt. Det enda som återstår är att skriva koden för det som skall utföras då de olika objekten aktiveras. Dessa objekt kan bestå av:

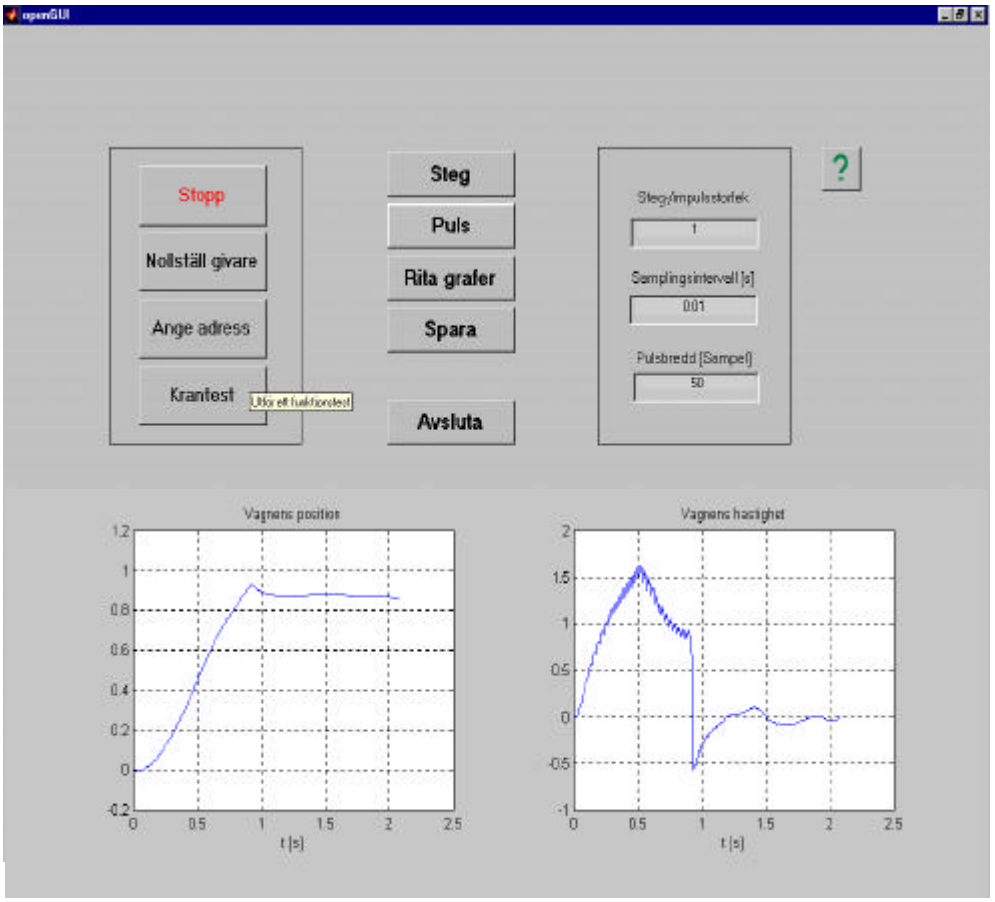
- pushbutton → tryckknapp
- edit → ruta för inmatning av tecken
- popup → ”Rullgardinsmeny”
- radiobutton/checkbox → rutor som antingen är markerade eller ej
- slider → rullningslist
- list box → ett eller flera alternativ kan väljas
- axes → graffönster
- text → statisk text, T.ex. titlar, axelbeskrivningar och liknande

se vidare i bilaga 4 för mer information om de olika verktygen samt exempel på hur dessa GUI kan byggas upp.

Några av de kommandon som finns för att utöka sitt gränssnitt med t.ex. filhantering, diverse dialogrutor redovisas nedan:

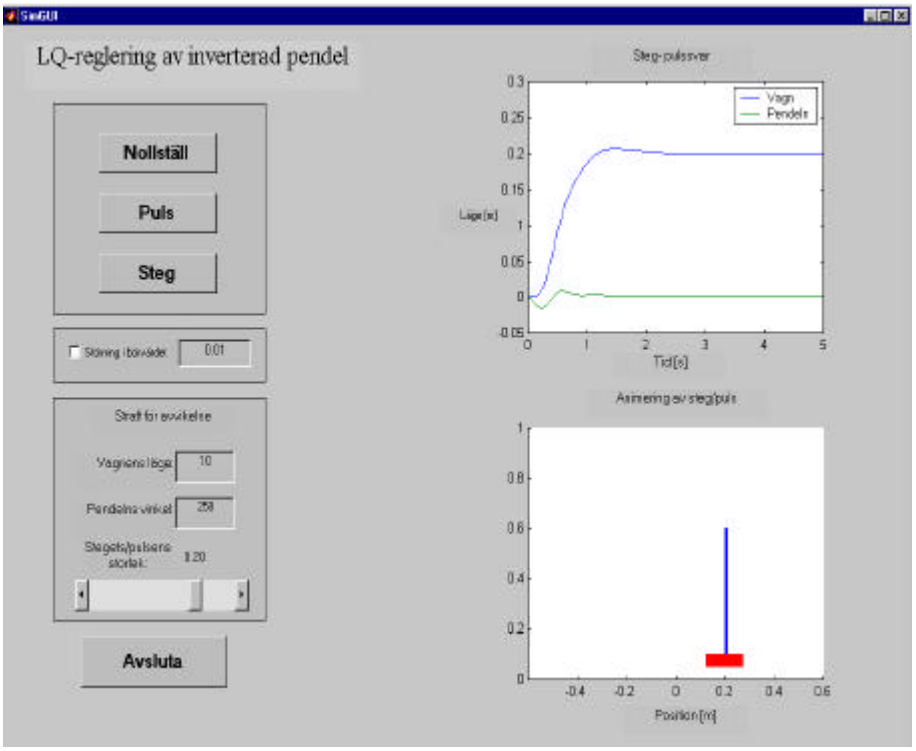
- uigetfile, öppnar en fil
- uiputfile, sparar fil
- warndlg, ger en ruta med ett varningsmeddelande
- errordlg, felmeddelande
- questdlg, dialogruta med max tre ”svarsknappar”

samt ett flertal andra alternativ som hjälprutor och mycket annat.
För att inhämta dessa kunskaper har boken ”Användarhandledning för MATLAB 5” [10] har varit till mycket stor hjälp.



Figur 14. *Det grafiska gränssnittet till openGUI*

I figur 14 redovisas det GUI som använts för att samla mätdata från reglerobjektet.



Figur 15. *Det grafiska gränssnittet till SimGUI*

Det GUI som redovisas i figur 15 har använts för att experimentera med LQ för att se hur olika straff påverkar pendeln och vagnens beteende.

9 Resultat/slutsatser

I stort har alla mål uppfyllts förutom att det trots flera försök inte gick att överföra mätdata till MATLAB på ett riktigt sätt då en snabbt varierande styrsignal behövdes. Efter upprepade försök att på olika sätt beräkna styrsignalens värde och överföra den till processen avbröts vidare försök i samråd med handledaren. I stället användes den modell i SIMULINK som tagits fram för att ”testa” regulatorn innan försöken inleddes med den fysiska pendeln. Det var med denna matematiska modell som försöken senare utfördes på. En kort beskrivning av linjärkvadratisk reglering kan sägas vara ”Det är inte straffen för varje individuellt tillstånd som är av största vikt utan deras inbördes förhållande”.

De andra målen har däremot uppfyllts till fullo. LQ-regleringens fördelar liksom nackdelar har klarlagts för författaren till denna rapport. Det kan konstateras att fördelarna med råge överväger de nackdelar som finns. De nackdelar som kan nämnas är vissa problem med robusthet då modellen av reglerobjektet inte är tillräckligt bra och

reglerobjektet finns i en ”brusig” miljö. Den största fördelen är just LQ-regleringens strafftänkande och att det är möjligt att snabbt ändra regulatorns egenskaper.

Det kan också konstateras att de i MATLAB inbyggda hjälpmedlen som är samlade i ”*Guide Control Panel*” för att utveckla grafiska användargränssnitt är mycket användbara och lättarbetade efter viss inlärningstid. Men det blir ofta mycket onödig kod som genereras ty många så kallade ”Properties” (kan jämföras med egenskaper) genereras som inte alls behövs. Detta kan dock undvikas på två sätt, dels kan all kod skrivas in för hand men det tar ofta lång tid då det blir flera rader för varje objekt(t.ex. en ”pushbutton”) dels kan m-filen ”rensas” från onödiga ”Properties”. Men det enklaste är trots allt att låta den onödiga koden finnas med då den inte har någon inverkan på resultatet. Det kan dock konstateras att ju mer grafik som använd desto snabbare måste datorn vara för att klara av att hinna skapa de olika objekten.

Slutligen kan det konstateras att examensarbetet varit en framgång om man bortser från problemen med att överföra mätvärden/styrsignal mellan MATLAB och pendeln. Den största behållningen har varit den fördjupade förståelse och kunskap om reglerteori samt en hel del kunskap om hur grafiska gränssnitt kan byggas upp.