Linköpings universitet
Institutionen för systemteknik (ISY)
Fordonssystem

Laborationskompendium Fordonsdynamik TSFS02

Linköping 2013

Innehåll

Kapitel 1

Laboration 3 - Anti-sladd

Ett system för anti-sladd har till uppgift att understödja föraren i styrningen under kritiska situationer. Vanligt förekommande beteckningar på sådana system är till exapmel ESP - electronic stability program, ESC - electronic stability control, DSC - dynamic stability control, AYC - active yaw control och VDC - vehicle dynamics control.

Syftet med systemet är att kompensera över- och understyrning genom att påverka fordonet med ett gir-moment, det vill säga ett vridande moment kring den vertikala axeln. Momentet skapas vanligen genom att bromsa individuella hjul. Det är även möjligt att skapa moment på andra sätt, genom exempelvis aktiv styrning av styrvinkel eller drivande moment.

Systemet är i ett fordon vanligen integrerad i en regulatorhierarki tillsammans med system för ABS (anti-lock braking system) och TRC (traction-control system). För de obligatoriska momenten i denna laboration fokuseras endast på ESP.

Laborationen utförs i simulatorn RACER. I appendix B finns information om hur simulatorn används.

Syfte

Laborationen ska ge insikt i hur ett system för anti-sladd fungerar och några av de utmaningar som uppstår vid design av ett sådant system.

1.1 Examinationskrav

För att bli godkänd på laborationen ska följande uppfyllas:

1. Uppfylla förkunskapskraven som anges i nästa avsnitt.
2. En godkänd skriftlig redogörelse där alla förberedelseuppgifter och övriga uppgifter besvaras och motiveras med hjälp av fullständiga resonemang och figurer med, till exempel, data från simulatorn. Deadline ges på kurshemsidan.
3. Alla uppgifter ska vara individuellt lösta av varje grupp.

1.2 Förkunskapskrav

För att få göra laborationen ska följande uppfyllas:

1. Förberedelseuppgifterna ska vara utförda innan laborationstillfället, och kunna redovisas vid laborationstillfället.
2. Göra sig förtrogen med databehandling och ritfunktioner i MATLAB.
3. Göra sig tillräckligt förtrogen med C++ för att skriva enklare kod.

Förberedelseuppgifterna kommer att kontrolleras vid laborationstillfället. I MATLAB behöver man kunna grundläggande kommandon för åtkomst av matriser och ritkommandon.
För programmering i C++ behöver man åtminstone känna till enkla nyckelord, variabeltyper, funktionsanrop och -deklarationer, operatorer och villkorssatser.

1.3 Förberedelseuppgifter

I avsnittet följer beskrivning av de uppgifter som ska utföras innan laborationstillfället. Observera att även dessa uppgifter ska redovisas i laborationsrapporten.

1.3.1 Design

Målet med ESP kan ses vara att minimera avvikelsen mellan ett av föraren önskat uppträdande och det verkliga uppträdandet. Kriterierna kan relateras till gir-hastighet Ω_z och body-slip β. En regulator för ett gir-moment ΔM kan då skrivas som

$$\Delta M = \Delta M\left(\beta^{nom} - \beta, \Omega_z^{nom} - \Omega_z\right) \tag{1.1}$$

där Ω_z^{nom} och β^{nom} är de nominella värden som motsvarar önskat uppträdande.

En proportionell reglering ger

$$\Delta M = k_1\left(\beta^{nom} - \beta\right) + k_2\left(\Omega_z^{nom} - \Omega_z\right) \tag{1.2}$$

där k_1 och k_2 är inställningsparametrar.

Uppgift 1

Bestäm lämpliga uttryck för Ω_z^{nom} och β^{nom} i (1.1). Ingående storheter ska förklaras i tydliga figurer.

Motivera ordentligt vilka antaganden som ligger bakom dessa uttryck, och varför ni har valt att beräkna de nominella variablerna på just detta vis.

Uppgift 2

Designa en regulator för ett gir-moment ΔM som ska påverka fordonet, ett förslag är givet i (1.2). Det finns nödvändiga givare och färdiga observatörer som ger följande signaler:

δ Styrvinkel (rad).

Ω_z Gir-hastighet (rad/s).

$\omega_{1,2,3,4}$ Rotationshastigheter för respektive hjul (rad/s).

$\dot{v}_y$ Tidsderivatan av lateralhastigheten, $\dot{v}_y = a_y - \Omega_z v_x$ (m/s^2).

Teckenkonventionerna är enligt kursboken samt följande:

$\delta > 0$ framhjulen ställda åt höger $\quad$ $\delta < 0$ framhjulen ställda åt vänster

Ange fullständiga uttryck för alla ingående storheter i regulatorn. De ska endast bero på insignaler, skattade variabler och parametrar.

Uppgift 3

Givet er design, vilka tecken ska förekommande regulatorkonstanter ha för att få ett rimligt beteende? Har ni valt en design enligt (1.2) så är det alltså tecknen för konstaterna k_1 och k_2 som ska bestämmas.

1.3.2 Aktuering

Det beräknade gir-momentet (1.1) kan aktueras på olika sätt. I laborationen kommer vi att studera en vanlig metod som innebär att hjul bromsas individuellt för att uppnå önskat vridande moment.

Om ett beräknat önskat moment ska kunna effektueras genom en kraft i kontaktytan mellan däck och väg måste aktuella förhållanden tillåta en kraft av tillräcklig storlek. Det är även så att när det longitudinella slippet ökar vid bromsning så minskar den laterala kraften vilket förändrar gir-momentet. I laborationen är det tillåtet men inte nödvändigt att ta hänsyn till dessa effekter.

Om trögheten hos hjulen slutligen försummas gäller då det enkla sambandet $M_b = r_w F_b$ där M_b är applicerat moment från bromsen (på hjulet), r_w är hjulradien och F_b är kraften som utvecklas i kontaktytan.

Uppgift 4

Antag att varje hjul kan bromsas oberoende och individuellt mellan ingen till full bromsverkan. Redogör för hur ΔM från regulatorn kan effektueras.

- Vid vilket eller vilka hjul ska en bromskraft appliceras och varför? Hur skiljer det sig mellan exempelvis vänster-, höger-sväng, över- och understyrning?
- Hur ska storleken på bromsmomentet M_b beräknas i de olika fallen?

Alla uttryck ska motiveras och härledas. Införda storheter ska förklaras i tydliga figurer.

Uppgift 5

Med avseende på de uttryck som tagits fram i tidigare uppgifter,

- gör en separat lista över eventuella tillståndsvariabler, och
- gör en separat lista över eventuella modellparametrar som behövs.

Använd bilen `Alfa Romeo GTA` och identifiera värden på nödvändiga parametrar, se appendix A.

1.4 Uppgifter

I avsnittet följer beskrivning av de uppgifter som ska utföras och rapporteras utöver förberedelseuppgifterna.

1.4.1 Implementering och utvärdering

Den designade regulatorn ska implementeras och utvärderas i simulatorn RACER. Koden ska skrivas i C++. Använd beskrivande variabelnamn och tydliga kommentarer. Otydlig kod kommer inte att beaktas. Se vidare i appendix C.

Tänk på hur regulatorn ska interagera med förarens kommandon som gas- och bromspedalläge. Vidare, om ett gir-moment beräknas enligt (1.2) kommer det sällan att vara exakt noll. Detta bör implementeringen ta hänsyn till. Det kan till exempel göras genom att införa tröskelvärden eller hysteres för aktiveringen av algoritmen.

Uppgift 6

Kopiera RACERtill erat konto, se appendix B.

Implementera er regulator enligt förberedelseuppgifterna. Utgå från filen `controller.cpp` i katalogen `src` vilken innehåller ett självförklarande programskelett att utgå från. Bifoga koden till er rapport. Försäkra er om att koden är väl indenterad och formaterad för att vara väl läsbar i utskriven form. Rimlig storleksordning på parametrarna k_1 och k_2 är 10^3. Exakta värden får testas fram.

Uppgift 7

Utvärdera regulatorn. Försäkra er först om att den fungerar som ni har tänkt.

Välj banan `Slippery`. Den är ovalt formad med hala avsnitt. Prova till exempel att göra ett byte från ena sidan av vägen till den andra på det hala området framför hindret. Gör det med ingångshastigheter om ungefär 40-50 km/h med och utan regulator. Prova även att köra i kurvan med samma ingångshastigheter. Använd gärna farthållaren för att få jämförbara försök.

Se i appendix B om handhavandet för att till exempel rita jämförande figurer och logga data.

Redovisa i rapporten åtminstone två körningar på det hala området med och utan regulator, en undanmanöver förbi hindret och en körning i kurvan. För en rättvis jämförelse, använd ungefär samma utgångshastighet, och motivera varför det syns att er regulator fungerar.

Extrauppgifter

- Reglera också motormomentet när regulatorn är aktiv.
- Ta fram och implementera begränsningar på börvärdena. Utvärdera.
- Bromsa ytterligare ett hjul om önskat girmoment inte uppnås med ett hjul.
- Hantera fallet då föraren bromsar samtidigt som regulatorn arbetar. Beskriv tillvägagångssätt och utvärdera.

Bilaga A

Laboration 3 - Parametrar

I tabellerna A.1 till A.5 anges ett urval av parametrar som gäller för den bakhjulsdrivna bilen `Alfa Romeo GTA` i simulatorn RACER. Position anges i (X, Y, Z) relativt koordinatsystemet i läroboken.

Parameter	Värde	Enhet
Karossbredd	1.58	m
Karosslängd	4.08	m
Karosshöjd	1.00	m
Massa, kaross	800	kg
Massa, motor	150	kg

Tabell A.1 *Fordonsdata*

Parameter	Värde	Enhet
Massa	15	kg
Max. moment fr. broms	750	Nm
Position fr. tyngdpunkt	(1.19,-0.62,-0.12)	m
Radie	0.32	m
Stationär camber	-0.75	°
Styrning	[-40,40]	°
Tröghet kring rot.axel	1	kgm^2
Toe-in	0.1	°
Sidkraftskoeff.	$9.35 \cdot 10^4$	N/rad

Tabell A.2 *Vänster framhjul och däck*

Parameter	Värde	Enhet
Massa	15	kg
Max. moment fr. broms	750	Nm
Position fr. tyngdpunkt	(1.19,0.62,-0.12)	m
Radie	0.32	m
Stationär camber	-0.75	°
Styrning	[-40,40]	°
Tröghet kring rot.axel	1	kgm^2
Toe-in	0.1	°
Sidkraftskoeff.	$9.35 \cdot 10^4$	N/rad

Tabell A.3 *Höger framhjul och däck*

Parameter	Värde	Enhet
Massa	15	kg
Max. moment fr. broms	500	Nm
Position fr. tyngdpunkt	(-1.41,-0.62,-0.12)	m
Radie	0.32	m
Stationär camber	-0.5	°
Styrning	0	°
Tröghet kring rot.axel	1	kgm^2
Toe-in	0.5	°
Sidkraftskoeff.	$9.35 \cdot 10^4$	N/rad

Tabell A.4 *Vänster bakhjul och däck*

Parameter	Värde	Enhet
Massa	15	kg
Max. moment fr. broms	500	Nm
Position fr. tyngdpunkt	(-1.41,0.62,-0.12)	m
Radie	0.32	m
Stationär camber	-0.5	°
Styrning	0	°
Tröghet kring rot.axel	1	kgm^2
Toe-in	0.5	°
Sidkraftskoeff.	$9.35 \cdot 10^4$	N/rad

Tabell A.5 *Höger bakhjul och däck*

Bilaga B

Handhavande RACER

B.1 Handhavande

I detta avsnitt beskrivs kort handhavandet av simulatorn RACER[1] och MATLAB för laborationen.

Alla kommandon, förutom matlab-kod, är tänkt att skrivas i ett terminalfönster.

B.1.1 Kopiera

Nödvändiga filer kopieras med kommandot (observera den avslutande punkten)

```
cp -r /site/edu/fs/TSFS02/Racer .
```

som placerar filerna i den aktuella katalogen i underkatalogen `Racer/`.

B.1.2 Kalibrera

När man väljer att köra med ratt kan den behöva kalibreras. Detta görs genom att köra programmet kcontrol. Skriv

```
kcontrol &
```

och gå in under `peripherals/joystick` samt klicka på `calibrate`. Det räcker att kalibrera de två första axlarna. Om fler efterfrågas räcker det att endast trycka på valfri knapp på ratten för att komma vidare. Studera värdet på efterfrågad axel som visas i

[1] Se www.racer.nl

nedre vänstra hörnet av fönstret. Gas- och bromspedalerna definieras till exempel på samma axel.

B.1.3 Köra

Simulatorn RACER startas med kommandofilen `runRacer.sh` i katalogen `Racer/`. Skriv

```
cd Racer
./runRacer.sh
```

I simulatorn finns självförklarande menyer för att välja bil, bana samt för att starta. Se till att inte `num-lock` är intryckt under körning, då fungerar inte snabbtangenterna. En körning avslutas med `esc`.

I filen `Racer/racer.ini` kan man välja styrdon mellan tex ratt, mus och tangentbord, se kommentarer i filen.

Några användbara kortkommadon:

`C` Kopplar i och ur farthållaren. Använd +/- på det numeriska tangentbordet för att styra börvärdet. Status visas med text uppe till vänster i bild. Farthållaren kopplas automatiskt ur vid inbromsning.

`E` Kopplar i och ur den egengjorda regulatorn. Filen `controller.cpp` anropas första när regulatorn kopplas in.

`L` Startar/avslutar loggning av data, se nedan.

`P` Paus.

`shift+R` Börja om från utgångspositionen på banan.

`Esc` Avsluta.

`1,2,3,4...` Byter kamera vinkel.

`Num-Pad` Flyttar runt kameran. Observera att `num-lock` måste vara släckt.

Om motorn stannar så kan den startas med den övre högra knappen på ratten.

B.1.4 Logga och bearbeta data

Genom att trycka `L` i RACER startas en loggning. Vid nästa tryck på tangenten sparas en MATLAB-fil[2] med data. Notera att denna fil skrivs över vid varje ny loggning. Ladda in data i MATLAB genom att använda `LoadRacerData` (se nedan), och spara det till en fil i er hemkatalog, använd kommandot `save`. För att bearbeta data, öppna

[2] `/tmp/logger.mat`

först filen `LoadRacerData` och läs förklaringarna för alla tillgängliga variabler. Alla script ligger i katalogen `Racer/matlab/`.

Följande script i MATLAB är användbara och tillhandahålls som exempel för att underlätta laborationen.

`LoadRacerData` Läser in loggad data.

`ESPplot` Ritar, i figur 1, några intressanta storheter för laborationen Anti-sladd.

`Suspensionplot` Ritar, i figur 1–3, några intressanta storheter för laborationen Semi-aktiv dämpning. Se `help Suspensionplot`.

Alla script kan anropas flera gånger för att göra jämförande figurer.

Ett exempel på arbetsgång är således att efter en körning i RACER växla till MATLAB och skriva

```
>> LoadRacerData
>> save name.mat
```

där `name` är en lämplig beskrivning. För att göra en jämförande figur mellan två körningar `one` och `two`, skriv

```
>> load one.mat
>> ESPplot
>> load two.mat
>> ESPplot
```

eller

```
>> Suspensionplot('one.mat','the first one')
>> Suspensionplot('two.mat','the second one')
```

Bilaga C

Programmering

Er regulator ska implementeras i filen `controller.cpp`. Använd den givna mallen för respektive laboration, den innehåller en del kommentarer och är delvis självförklarande. Filen `controller.cpp` ska ligga i katalogen `Racer/src`. Använd till exempel `emacs` för att ändra. Skriv

```
cd Racer/src
emacs controller.cpp &
```

För att kompilera koden används en så kallad make-fil som läses in genom att köra kommandot `make`. Skriv

```
cd Racer/src
make
```

Om kompileringen lyckas kan RACER startas och er regulator kommer att användas. I simulatorn aktiveras och deaktiveras regulatorn med tangenten `E`. Uppe i vänstra hörnet anges om regulatorn är aktiverad. Se till att inte `num-lock` är intryckt under körning, då fungerar inte snabbtangenterna.

C.1 Gränssnitt

Huvudfunktionen för regulatorn är
`void Controller(const ControlInput& In, ControlOutput& Out)`
där argumenten är objekt som motsvarar insignaler respektive utsignaler.

Medlemsfunktioner

Koden för att hämta tillgängliga insignaler är redan skriven. För att sätta utsignaler och realisera er regulator finns en uppsättning medlemsfunktioner till objekt av klassen `ControlOutput`.

`void SetPedalLevel(int level)`
Sätter pedalnivån till `level`. Nivån kan ses som normaliserat motormoment och ligger i intervallet [0,1000].

`void SetBrakeLevel(int level)`
Sätter bromsnivån till `level`. Nivån kan ses som normaliserat bromsmoment och ligger i intervallet [0,1000].

```
void SetBrakeFactors(int FrontLeft, int FrontRight,
                     int RearLeft, int RearRight)
```
Sätter en skalning av bromsnivån på respektive hjul. Skalfaktorerna är ett tal i intervallet [0,100].

```
void SetDampFactors(float FrontLeft, float FrontRight,
                    float RearLeft, float RearRight)
```
Sätter dämpfaktorerna på respektive hjulupphängning.

`bool SetMonitor(int i, float value)`
Sätter monitor i till givet värde i $value$. Returvärdet är sant om $i \in \{0, \ldots, N_m - 1\}$ där N_m är antalet monitorer, annars falskt. För närvarande är $N_m = 4$.

Medlemsvariabler

Det finns vidare två medlemsvariabler i objekt av klassen `ControlOutput` som kan ändras.

`int State`
Anger regulatorns tillstånd. Bör sättas till en av flaggorna `INACTIVE`, `ESP_ACTIVE`, `ASC_ACTIVE`. När `State` är skilt från `INACTIVE` skrivs det ut ett meddelande på skärmen i RACER. Dessa kan ses som de lampor som brukar lysa upp på instrumentbrädan. Förkortningarna står här för Electronic Stability Program och Active Suspension Control.

Exempel, ESP

För att bromsa höger framhjul 50% av kapaciteten kan alltså följande kod användas

```
Out.SetBrakeLevel(1000);
Out.SetBrakeFactors(0,50,0,0);
```

eller ekvivalent

```
Out.SetBrakeLevel(500);
Out.SetBrakeFactors(0,100,0,0);
```

För att indikera att er regulator arbetar, skriv
`Out.State = ESP_ACTIVE;`

För att sätta monitor nummer 2 till 10, skriv
`Out.SetMonitor(1,10);`

Exempel, ASC

För att sätta dämpfaktorerna till 2100 Ns/m i fram och 1800 Ns/m i bak, skriv

```
Out.SetDampFactors(2100,2100,1800,1800);
```

Funktionen sätter samma koefficienter för expansion och kompression hos dämparen.

C.2 Vanliga C++ kunskapsluckor

Observera att enligt förkunskapskraven 1.2 är det upp till studenten själv att fylla eventuella kunskapsluckor inom C++ innan labbtillfället. Här följer ändå något om de vanligast luckorna.

Deklaration

Innan en variabel används måste den deklareras. Det måste specifieras av vilken typ den ska vara.

```
int foo;    //  en heltalsvariabel (eng. integer)
float bar;  //  en flyttalsvariabel
bool isFoo; //  en boolsk variabel
```

Det finns fler variabeltyper men dessa borde räcka för laborationen.

Static

Filen `Controller.cpp` som regulatorn implementeras i består av en funktion (`Controller`) som anropas upprepade gånger från en yttre loop. Efter avslutat funktionsanrop försvinner alla variabler som skapats i funktionen. Om ett värde önskas behållas mellan två funktionsanrop måste den deklareras som `static`. T ex

```
static int foo;
float bar;
```

I ovan kommer alltås värdet på `foo` att finnas kvar nästa gång funktionen anropas medan värdet på `bar` kommer att försvinna.

Omtypning

En funktion som vill ha en `integer` som argument kommer inte att ta emot en `float`. Det vill säga funktionen

```
void foobar(int arg1)
```

kommer inte att acceptera anropet `foobar(bar)` men `foobar(foo)` går bra. Om det ändå är värdet på `bar` som önskas användas kan `bar` typas om,

```
foobar((int)bar)
```

Notera dock att decimalerna i `bar` kapas utan avrundning.