Linköpings universitet
Institutionen för systemteknik (ISY)
Fordonssystem

Laborationskompendium Fordonsdynamik TSFS02

Linköping 2013

Innehåll

Kapitel 1

Laboration 4 - Semi-aktiv dämpning

Syfte

Laborationen ska ge insikt i principerna för design av ett system med semi-aktiva dämpare.

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 detta.
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.

Det förutsetts givetvis att kursen är välinhämtad. Speciellt kapitel 7.2 och 7.4 i Wong.

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

I ett semi-aktivt system för hjulupphängningen finns typiskt en vanlig fjäder tillsammans med en dämpare där dämpkoefficienten kan regleras. Karakteristiken hos upphängningen kan därmed ändras under färd.

Med aktiv reglering kan till exempel rörelsen i roll-, gir- och tippled kontrolleras i olika körfall. Målen kan vara komfort-, säkerhets- eller prestandaorienterade som att uppnå isolation från ojämnheter i vägen eller att maximera väghållning och styregenskaper.

I den här laborationen fokuseras på komfort och specifikt på isolation av skarpa ojämnheter i vägen.

Uppgift 1

Rita en figur för en kvartsbilmodell och inför nödvändiga parametrar och koordinater. Använd sedan dessa beteckningar i senare uppgifter och i er implementation.

Uppgift 2

I läroboken tas två reglerstrategier upp. Formulera dessa matematiskt med hjälp av de beteckningar ni infört tidigare.

Uppgift 3

Vi antar här, något förenklat, att det finns nödvändiga givare och filter som ger följande signaler.

a_i Acceleration i höjdled av fjädrad massa vid hjul i (m/s^2) (positiv nedåt).

p_i Total längd av fjäder i (m).

I implementationsmiljön finns även en tidsreferens tillgänglig.

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

Med avseende på de uttryck som tagits fram,

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

Glöm inte att förse alla värden med fysikalisk enhet. Använd bilen `BMW Mini Cooper` för att identifiera nödvändiga parametrar, se appendix A.

Notera att p_i är den totala längden. Därför behöver de längder som motsvarar statiskt jämvikt, $p_{i,0}$, beräknas.

1.3.2 Utvärdering

Målet med systemet kan vara mångfacetterat, som nämnts ovan. I läroboken tas tre kvantitativa kriterier upp. Dessa är respektive rms-värde av följande storheter.

J_1 : accelerationen hos den fjädrade massan

J_2 : avståndet mellan ofjädrad och fjädrad massa

J_3 : avståndet mellan vägbanan och den ofjädrad massan

$$J_1 = \sqrt{\frac{1}{T_1 - T_0}\int_{T_0}^{T_1} \ddot{z}_1^2 dt}$$

$$J_2 = \sqrt{\frac{1}{T_1 - T_0}\int_{T_0}^{T_1} L_i^2 dt}$$

$$J_3 = \sqrt{\frac{1}{T_1 - T_0}\int_{T_0}^{T_1} (z_0 - z_2)^2 dt}$$

Uppgift 4

Svara på följande frågor angående kriterierna

a) Vad är ett rms-värde och vad innebär det?

b) Vad är dessa kriterier mått på, i ord som betyder något för en förare?

c) Hur påverkas de av förändrad dämpning? (Tips: Studera figur 7.13, 7.16 och 7.19 i boken.)

d) Nämn en fundamental avvägning bland dessa mål funktioner (J_1, J_2, J_3), och motivera med hänvisningar till figur 7.13, 7.16 och 7.19 i boken.

1.4 Uppgifter

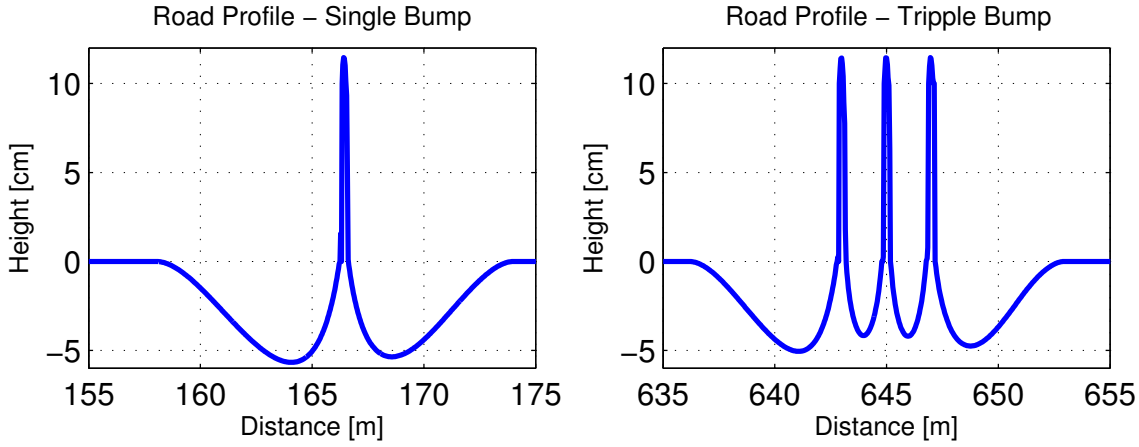
I avsnittet följer beskrivning av de uppgifter som ska utföras och rapporteras utöver förberedelseuppgifterna.

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 appendix B för information om handhavandet av simulatorn.

Uppgift 5

Först ska olika inställningar av passiva dämpare provas. Välj banan `Bumpy`. Den är ovalt formad med skarpa gupp på långsidorna. Guppens vägprofiler kan ses i figur 1.1. Använd bilen `BMW Mini Cooper`. Se vidare i appendix B om handhavandet för att till exempel rita jämförande figurer.



Figur 1.1 *Vägprofilerna för de två guppen på* `Bumpy`*. Till vänster enkelguppet på startsträckan och till höger trippelguppet på den andra raksträckan.*

Utför och logga körningar med endast passiv dämpning. Bestäm *en* hastighetsnivå att utföra testerna vid, ungefär 30 km/h kan vara lämpligt. Använd farthållaren för att få samma hastighet vid de olika försöken. Kör över det ensamma guppet och använd följande inställningar.

- c_{soft} för alla dämpare.
- Standardinställningen.
- c_{firm} för alla dämpare

Se appendix A för förklaring av $c_{\texttt{soft}}$ och $c_{\texttt{firm}}$.

Använd division- och produkt-tangenterna på det numeriska tangentbordet för att ändra dämparnas inställning i RACER. I övre vänstra hörnet av skärmen finns en del information, bland annat aktuella värden på koefficienterna och farthållarens börvärde.

Värdena på kriterierna 1–3 ovan skrivs ut i figurerna av `Suspensionplot`. I laborationen fokuserar vi på komfort vilket innebär att vi vill ha ett lågt värde på J_1. Hur förändras J_1 vid olika inställningar och varför? Finns det fler utmärkande och intressanta skillnader med de olika inställningarna?

Spara dessa körningar och använd som referens för era regulatorer.

Uppgift 6

Implementera regulator-strategin *skyhook* enligt förberedelseuppgifterna. Utgå från filen `skycontroller.cpp` i katalogen `src`. Spara den som `controller.cpp` i samma katalog (kom ihåg att först spara undan implementationer från tidigare laborationer som ska sparas). Filen 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.

Uppgift 7

Utvärdera regulatorn. Försäkra er först om att den fungerar som ni har tänkt. Beskriv främst med hjälp av figurer från loggad data men även med värdet på kriteriet J_1 hur era regulatorer står sig jämfört med passiva dämpare.

Redovisa i rapporten resonemang tillsammans med figurer som visar att er regulator fungerar. Rita i dessa figurer även körningar (i samma hastighet) utan reglering för jämförelse, se appendix B. Nödvändigt krav för laborationen är att redovisa körningar på det ensamma guppet i den hastighetsnivå ni valt.

Extrauppgifter

- Implementera den kontinuerliga reglerstrategin från kursboken.
- Utvärdera på ojämnheten som består av tre skarpa gupp.
- Studera prestandan vid högre hastighetsnivåer.

Bilaga A

Laboration 4 - Parametrar

I tabellerna A.1 till A.5 anges ett urval av parametrar som gäller för bilen `BMW Mini Cooper` i simulatorn RACER. Position anges i (X, Y, Z) relativt koordinatsystemet i läroboken. De semi-aktiva dämparna har alla arbetsområdet $[c_{soft}, c_{firm}] = [800, 5000]$ Ns/m.

Tabell A.1 *Fordonsdata*

Parameter	Värde	Enhet
Karossbredd	1.73	m
Karosslängd	3.63	m
Karosshöjd	1.31	m
Massa, kaross	948	kg
Massa, motor	80	kg

Tabell A.2 *Vänster framhjul*

Parameter	Värde	Enhet
Massa	18	kg
Position fr. tyngdpunkt	(0.93,-0.75,-0.25)	m
Radie	0.30	m
Fjäderlängd (obelastad)	55	cm
Spelrum för fjäder	[35, 60]	cm
Fjäderkonstant	$32 \cdot 10^3$	N/m

Tabell A.3 *Höger framhjul*

Parameter	Värde	Enhet
Massa	18	kg
Position fr. tyngdpunkt	(0.93,0.75,-0.25)	m
Radie	0.30	m
Fjäderlängd (obelastad)	55	cm
Spelrum för fjäder	[35, 60]	cm
Fjäderkonstant	$32 \cdot 10^3$	N/m

Tabell A.4 *Vänster bakhjul*

Parameter	Värde	Enhet
Massa	18	kg
Position fr. tyngdpunkt	(-1.53,-0.75,-0.25)	m
Radie	0.30	m
Fjäderlängd (obelastad)	53	cm
Spelrum för fjäder	[30, 55]	cm
Fjäderkonstant	$26 \cdot 10^3$	N/m

Tabell A.5 *Höger bakhjul*

Parameter	Värde	Enhet
Massa	18	kg
Position fr. tyngdpunkt	(-1.53,0.75,-0.25)	m
Radie	0.30	m
Fjäderlängd (obelastad)	53	cm
Spelrum för fjäder	[30, 55]	cm
Fjäderkonstant	$26 \cdot 10^3$	N/m

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 Ladda in rätt MATLAB-version

Nyare versioner av MATLAB är ej kompatibla med RACER, och därför måste en äldre MATLAB-version "laddas in" innan RACER startas. Detta görs genom att ni skriver in följande kommando i terminalen, innan ni kör igång RACER första gången:

```
module load matlab/7.9
```

Så länge ni inte stänger ner eller byter terminal-fönster behöver ni bara köra module-kommandot en gång.

B.1.4 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.1.5 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 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 Semiaktiv 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')
```

[2] `/tmp/logger.mat`

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 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.