Linköpings tekniska högskola, ISY, Fordonssystem

Laborationskompendium Fordonsdynamik TSFS02

Linköping 2006

Innehåll

Kapitel 1

Laboration 1 - ABS

Syfte

Laborationens syfte är att ge insikt i hur ett ABS-system fungerar och vilka avvägningar som är viktiga när man designar ett ABS-system.

1.1 Laborationsbeskrivning

Den bakomliggande grundtanken med denna laboration är att ge en känsla för vad ett ABS-system gör samt vilka förväntningar man kan ha på ett ABS-system. Detta ska uppnås genom att själv implementera och testköra en regulator och jämföra denna med enklare regulatorer samt det helt oreglerade fallet under olika vägbetingelser.

1.2 Examinationskrav

I detta avsnitt finns ett antal examinationskrav samlade. För att bli godkänd på laborationen måste man

1. Närvara vid laborationstillfället
2. I förväg ha sett till att man är förtrogen med MATLAB/SIMULINK. Man bör t.ex.
   - kunna koppla upp ett enkelt återkopplat system, simulera detta, exportera data till *Workspace* samt plotta resultatet.
   - ha läst igenom dokumentationen till simulinks relä- och integrator-block, speciellt om *Resetting the state* och *Defining initial conditions*.

(`doc simulink/integrator` i Matlab alternativt On-Line, se länk på hemsidan.)

3. Svara på alla obligatoriska frågor med fullständiga svar, ev med referenser till plottar.

4. Vid redovisningen kunna svara på några kontrollfrågor från assistenten.

1.3 Genomförande

Alla uppgifter i Laborationen förutsätter att du har tagit hem filerna på kurshemsidan och sparat ned dem i en Kurskatalog samt startat MATLAB och flyttat dig till din Kurskatalog. Du bör ha laddad hem följande filer

- `ABSController.mdl`
- `mkPlots.m`
- ...

Du bör även ha startat ABS-styrutrustningen genom att slå på den på knappen på baksidan, dra ur nödstoppsknappen om den är intryckt och trycka på den röda knappen på frontpanelen. ABS-styrutrustningen skall nu surra svagt.

Uppgift 1
Börja med att öppna modellen `ABSController` genom att antingen skriva
`>> ABSController`
vid prompten eller genom att klicka `File->Open` och sedan välja `ABSController.mdl`. Bygg sedan modellen genom att välja `Tools->Real Time Workshop->Build` alternativt genom att trycka `Ctrl-B`.

Öppna alla `Scope`-block som finns i modellen innan någon Simulering startas. Eftersom det är en realtidsmodell kommer SIMULINK nämligen inte att visa några signaler om inte blocken varit öppna under simuleringen. Data loggas dock till `Workspace` ändå.

Testkör sedan modellen genom att välja `Connect to target` i `Simulation` menyn i SIMULINK. Tryck sedan på *Play* symbolen eller välj motsvarande under `Simulation` menyn.

ABS utrustningen kommer nu att genomföra ett experiment med den inbyggda relästyralgoritmen. När experimentet är klart är det dags att titta på resultatet i *scope*-blocken.

Relä-styrningen fungerar enligt följande:

1. När bromsfasen av experimentet börjar ställs en förutbestämd konstant bromssignal ut, t.ex. maximal bromskraft.

2. När slip-värdet blir större än en viss tröskel ställs en annan konstant bromssignal ut, t.ex. noll bromskraft.
3. När slip-värdet bli mindre än en viss annan tröskel så återupptas bromsningen igen. Algoritmen börjar sedan om igen från steg 2.

Ungefär hur lång tid tog det att stanna?

Svar:

Ungefär hur lång blev stoppsträckan?

Svar:

Uppgift 2

SIMULINK har under simuleringen sparat data som `Scope`-blocken samlat upp i variabeln `ABSDataLog`. Spara detta data i en structvariabel genom att skriva

```
>> DataStruct = [];
>> DataStruct(1).Data = ABSDataLog
>> DataStruct(1).ExpName = 'Relay controller, dry conditions'
```

Innehållet i `DataStruct` kannu plottas med kommandot

```
>> mkPlots([1],{'r--'},[2 ]);
```

Kommandot ovan plottar innehållet i `DataStruct(1)` med en röd streckad linje med bredd 2. Det första argumentet i funktionen `mkPlots.m` bestämmer alltså vilka data som ska plottas, det andra argumentet i bestämmer linjetyp medans det tredje bestämmer vilken linjebredd respektive plot skall ha. I MATLAB-hjälpen kan man bl.a. läsa att följande linjetyps-kombinationer finns:

```
Various line types, plot symbols and colors may be obtained with
PLOT(X,Y,S) where S is a character string made from one element
from any or all the following 3 columns:

      b     blue          .     point              -     solid
      g     green         o     circle             :     dotted
      r     red           x     x-mark             -.    dashdot
      c     cyan          +     plus               --    dashed
      m     magenta       *     star             (none)  no line
      y     yellow        s     square
      k     black         d     diamond
                          v     triangle (down)
                          ^     triangle (up)
                          <     triangle (left)
                          >     triangle (right)
                          p     pentagram
                          h     hexagram
```

Uppgift 3
Det är nu dags att jämföra relä-styrningen med en tvärnit. Ställ om de båda övre väljarna i `ABSController` så att Relä-styrningen får en nolla och Hjullåsnings-styrningen får en etta. Gör sedan om simuleringen. Modellen behöver inte byggas om.

Spara därefter undan datat från simuleringen på position 2 i din datastruct
```
>> DataStruct(2).Data = ABSDataLog
>> DataStruct(2).ExpName = 'Wheel lock, dry conditions'
```
Plotta ånyo innehållet i `DataStruct` men den här gången med kommandot
```
>> mkPlots([1 2],{'r--','b-'},[2 2]);
```
Vad blev stoppsträckan?

Svar:
_____

Vilken av experimenten gav kortast stoppsträcka?

Svar:
_____

Uppgift 4
För att se om resultaten i Uppgift 3 blir annorlunda på blött underlag skall nu experiment med blött hjul göras. Utför därför två simuleringar med blött hjul och spara undan datat på samma sätt som ovan fast på plats 3 och 4. Döp t.ex. datat till *Relay controller, wet conditions* och *Wheel lock, wet conditions*

Plotta därefter på samma sätt som ovan de två experimenten i samma figur. Använd t.ex. kommandot
```
>> mkPlots([3 4],{'r--','b-'},[2 2]);
```

Vilken av experimenten gav kortast stoppsträcka?

Svar:
_____

Jämför experimenten med de två med torra förhållanden. Använd t.ex. kommandot
```
>> mkPlots([1 2 3 4],{'r--','b--','r-','b-'},[2 2 2 2]);
```

Vilken är den största fördelen med ett ABS bromssystem jämfört med att bromsa själv?

Svar:
_____

Uppgift 5
En nackdel med Relä-styrningen är att maximal bromskraft läggs ut direkt och låser hjulet i princip omgående, när bromsen släpper accelereras hjulet upp till bilens hastighet igen. Det mesta av bromsverkan sker då när hjulet accelereras upp igen och inte när man bromsar. Detta märks speciellt tydligt på det blöta underlaget ovan.

En bättre lösning vore att sakta öka bromskraften tills dess att hjulet låses och sedan snabbt *dra av* igen när hjulet låses.

Det är därför dags att implementera en egen ABS-variant. I blocket `Ramp Controller` finns början till ett ABS-system som *rampar* upp bromstrycket till ett bestämt värde istället för att lägga ut max bromskraft direkt. Funktionen skall vara följande:

1. Bromskraften rampas upp från noll tills det att slippet uppnår ett definierat värde. (Lutningen på rampen betecknas α)
2. Vid max-slip dras bromskraften ned till ett säkert värde. (Detta värde betecknas β)
3. När slippet kommer inom gränserna igen så rampas bromskraften ånyo upp.

Färdigställ den beskrivna metoden och provkör det på blött underlag. `Ramp Controller`-blocket är maskat och går tyvärr inte att dubbelklicka sig in i. Man får istället högerklicka på det och välja *Look under mask*. Glöm inte att bygga om modellen innan du provkör den.

Tips: En ramp kan t.ex. implementeras som en integration (”`doc simulink/integrator`”) av en konstant. Relä-funktionaliteten i Relä-styralgoritmen (”`doc simulink/relay`”) passar även i Ramp-styralgoritmen. Ni bör kunna lösa uppgiften med de redan befintliga blocken samt 3-5 ytterligare block.

Spara på samma sätt som ovan undan datat och plotta tillsammans med de övriga experimenten från blött underlag.

Lyckas den nya varianten bättre än den gamla?

Svar:

Finns det någon uppenbar nackdel med att rampa upp bromstrycket istället för att lägga ut maxtryck direkt?

Svar:

Uppgift 6
Utvärdera slutligen alla experimenten. Vad är det för skillnader mellan de olika reglerstrategierna och vilka är för- respektive nackdelarna (jämför exempelvis slippet mellan de olika experimenten)?

Svar:

Kapitel 2

Laboration 2 - Handling

I denna laboration ska beteende såsom över och understyrning studeras. Laborationen utförs i simulatorn RACER[1]. Till förfogande finns fyra bilar, Car I, Car II, Car III och Car IV. Dessa bilar är identiska vad gäller chassikonstruktion och hjulupphängningar, men det sitter olika däcksuppsättningar på dem. Viktfördelningen fram/bak är 50/50. För ytterliggare data över bilen, se Appendix B. Bilarna är bakhjulsdrivna.

Resultaten på varje uppgift ska redovisas i en skriftlig laborationsredogörelse. Alla resultat ska motiveras med plottar av variabler såsom styrvinkel, yaw-hastighet, longitudinell hastighet, bilens position osv.

Efter varje körning kan scriptet `[ESPplot]` användas för att plotta några intressanta variabler. Se även Appendix C för övriga tips om RACERoch Matlab.

2.1 Studie av stationärt beteende

Tips: Vid studier av en bils stationära beteenden vid tex olika hastigheter är det viktigt att accelera/decelera försiktigt mellan dessa hastigheter för att undvika transienta effekter. Tänk också på att förekomsten av longitudinellt slip påverkar de laterala krafterna.

1. Kör bilarna på bana "cirkel 195". Denna bana är en cirkel med radie ca 195 m. Ange respektive bils egenskaper och motivera din slutsats. I denna första uppgift ska ingen teoretisk analys göras, endast känslan får avgöra. Hur skiljer sig tex bilens beteende vid hastigheterna 10, 20, 40, 60, 80 och 100 km/h.
2. Till er hjälp finns de laterala däckskaraktäristikerna som .mat-filer (Car_A.mat, Car_B.mat, Car_C.mat och Car_D.mat). I dessa filer finns data för enskilda hjul.

[1] Se www.racer.nl.

Anta att båda hjulen på samma axel har samma karaktäristik. Ladda dessa data och och plotta Fy/Fz (normaliserad lateral kraft) mot alpha (slipvinkel här given i grader). Skissa eller plotta nu handlingdiagram för respektive däcksuppsättning. Markera ut i vilka områden bilen över- resp understyr och vart den är neutralstyrd. Visa också hur rattvinkeln ändras som funktion av hastighet när man kör på banan "cirkel 195". Para nu ihop däcksuppsättningarna (A, B, C och D) med respektive bil (I, II, III och IV).

3. Skissa även utifrån mätningar yaw velocity gain som funktion av hastighet för de olika bilarna.

4. Genomför konstant radie test och plotta styrvinkel mot lateral acceleration för respektive bil. Markera ut över-, under- och neutralstyrning. För detta ändamål kan banan "circles" vara användbar.

5. Genomför konstant styrvinkel test för respektive bil. Plotta styrvinkel mot lateral acceleration för respektive bil. Markera ut över-, under- och neutralstyrning och eventuellt instabilitetsområde.

6. På bana Carlswood NT finns en dragstrip. Testa där bilarnas stabilitet vid körning rakt fram. Stämmer tex den kritiska hastigheten som fås ur yaw velocity gain grafen?

2.2 Transient beteendestudie

Nu ska några transienta beteenden studeras. På dessa uppgifter behövs ingen motivering ges i form av plottar.

1. Kör den bil som är neutralstyrd på bana "cirkel 195" i ca 110 km/h eller på bana "circle 125" i ca 90 km/h. Försök hitta en stationäritet vid denna hastighet. När detta är gjort så släpp gasen hastigt med bibehållen rattvinkel. Vad händer? Vad beror detta på?

2. Hur ändras den neutralstyrda bilens egenskaper under svag acceleration? Vad beror detta på.

3. Kör någon egen vald bil på banan Carlswood NT och studera hur bilens beteende förändras under transienter såsom tex uppförs/nedförsbackar, acceleration/inbromsning, förändring av kurvradie etc.

4. Kör den understyrda bilen på Carlswood NT. Vad händer om man bromsar lite och svänger in i en kurva? Vad händer om man ger lite ökad gas ut ur en kurva? Vad händer om man bromsar hårt respektive ger full gas?

5. Slutligen, för att få bra känsla för de olika egenskaperna hos respektive bil, fri körning med alla bilar!

Kapitel 3

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 ESP - electronic stability program, 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.

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.

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.

3.1 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.

3. Alla uppgifter ska vara individuellt lösta av varje grupp.

3.2 Förkunskapskrav

För att få göra laborationen ska följande uppfyllas.

1. Förberedelseuppgifterna ska vara utförda innan laborationstillfället. Dessa ska vara nedskrivna i en början till laborationsrapport som tas med vid laborationstillfället.
2. Göra sig förtrogen med databehandling och ritfunktioner i MATLAB.
3. Göra sig tillräckligt förtrogen med C/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/C++ behöver man åtminstone känna till nyckelord, variabeltyper, funktionsanrop och -deklarationer, operatorer och villkorssatser.

3.3 Förberedelseuppgifter

I avsnittet följer beskrivning av de uppgifter som ska utföras och införas i en rapport innan laborationstillfället.

3.3.1 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 $\dot{\Psi}$ och body-slip β. En regulator för ett gir-moment ΔM kan då skrivas som

$$\Delta M = \Delta M \left(\beta_{nom} - \beta, \dot{\Psi}_{nom} - \dot{\Psi} \right) \tag{3.1}$$

där $\dot{\Psi}_{nom}$ och β_{nom} är de nominella värden som motsvarar önskat uppträdande.

En proportionell reglering ger

$$\Delta M = k_1 \left(\beta_{nom} - \beta \right) + k_2 \left(\dot{\Psi}_{nom} - \dot{\Psi} \right) \tag{3.2}$$

där k_1 och k_2 är inställningsparametrar.

Uppgift 1

Bestäm lämpliga värden på $\dot{\Psi}_{nom}$ och β_{nom} i (3.1). Alla uttryck ska motiveras och härledas. Införda storheter ska förklaras i tydliga figurer.

Uppgift 2

Designa en regulator för ett gir-moment ΔM som ska påverka fordonet, ett förslag är givet i (3.2). Följande insignaler finns tillgängliga.

δ Styrvinkel [rad].

$\dot{\Psi}$ Gir-hastighet [rad/s].

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

a_{lat} Lateral acceleration [m/s^2].

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

Uppgift 3

Om designen valts enligt (3.2), vilka tecken ska respektive regulatorkonstant k_1 och k_2 ha för att få en rimligt beteende? Är designen annorlunda än (3.2), bestäm eller gör en uppskattning av storleksordningen på eventuella regulatorparametrar.

3.3.2 Aktuering

Det beräknade gir-momentet (3.1) 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.

En förenkling kan göras genom att anta att maximal longitudinell kraft vid varje hjul är konstant. En mer avancerad metod kan till exempel ta hänsyn till dynamiken i kontakten mellan väg och däck genom att skatta vägens friktionstal och däckens slipvinklar.

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 kraften beräknas i olika fall?

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 `Car_III` och identifiera även värden på nödvändiga parametrar, se appendix A och B.

3.4 Uppgifter

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

3.4.1 Implementering och utvärdering

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

Tänk på hur regulatorn ska interagera med förarens kommandon som gas- och bromspedalläge. Vidare, om ett gir-moment beräknas enligt (3.2) kommer det sällan att vara exakt noll. Detta kan vara nödvändigt att ta hänsyn till i implementeringen. Det kan till exempel göras genom att införa tröskelvärden eller hysteres för aktiveringen av algoritmen.

Uppgift 6

Implementera er regulator enligt förberedelseuppgifterna. Utgå från filen `controller.cpp` i katalogen `racer0.5.0/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.

Uppgift 7

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

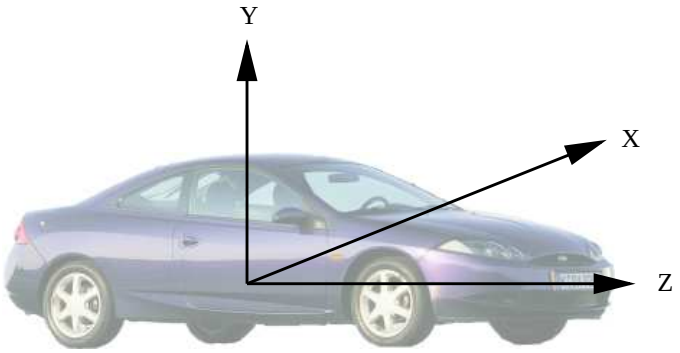
Banan `Carlswood NT - slippery` är en kopia av `Carlswood NT` där vägytan i depå-området har en kraftigt nedsatt friktion. På den raka vägsträckan med vägkoner, prova att göra ett filbyte från ena sidan om den dubbelstreckade mitten till den andra. Prova det med utgångshastigheter om ungefär 40,50,60,70 km/h med och utan regulator. Försök att gira mellan samma koner varje gång, det gör det lättare för er att rita jämförande figurer (se vidare i avsnittet Handhavande ovan).

Redovisa i rapporten åtminstone två körningar med och utan regulator vid samma utgångshastighet som visar att er regulator fungerar.

Bilaga A

Koordinatsystem

RACER använder sig av ett koordinatsystem med x-axeln ut genom förardörren, y-axeln upp genom taket och z-axeln ut genom framrutan, se figur A.1.



Figur A.1 *Koordinatsystemet*

Bilaga B

Parametrar

I tabellerna B.1 till B.5 anges ett urval av parametrar som gäller för bilarna CAR_I till CAR_IV i simulatorn RACER. Position anges relativt koordinatsystemet givet i föregående avsnitt.

Parameter	Värde	Enhet
Kaross, bredd	1.76	m
Kaross, längd	4.54	m
Kaross, höjd	1.24	m
Kaross, massa	870	kg
Motor, massa	180	kg

Tabell B.1 *Fordonsdata*

Parameter	Värde	Enhet
Massa	12	kg
Max. moment fr. broms	1100	Nm
Position fr. tyngdpunkt	(0.72,0.22,1.5)	m
Radie	0.35	m
Stationär camber	0	°
Styrning	[-40,40]	°
Tröghet kring rot.axel	1	kgm^2
Toe-vinkel	0	°

Tabell B.2 *Vänster framhjul*

Parameter	Värde	Enhet
Massa	12	kg
Max. moment fr. broms	1100	Nm
Position fr. tyngdpunkt	(-0.72,0.22,1.5)	m
Radie	0.35	m
Stationär camber	0	°
Styrning	[-40,40]	°
Tröghet kring rot.axel	1	kgm^2
Toe-vinkel	0	°

Tabell B.3 *Höger framhjul*

Parameter	Värde	Enhet
Massa	12	kg
Max. moment fr. broms	850	Nm
Position fr. tyngdpunkt	(0.72,0.22,-1.2)	m
Radie	0.35	m
Stationär camber	-1	°
Styrning	0	°
Tröghet kring rot.axel	1	kgm^2
Toe-vinkel	0	°

Tabell B.4 *Vänster bakhjul*

Parameter	Värde	Enhet
Massa	12	kg
Max. moment fr. broms	850	Nm
Position fr. tyngdpunkt	(-0.72,0.22,-1.2)	m
Radie	0.35	m
Stationär camber	-1	°
Styrning	0	°
Tröghet kring rot.axel	1	kgm^2
Toe-vinkel	0	°

Tabell B.5 *Höger bakhjul*

Bilaga C

Handhavande RACER

C.1 Handhavande

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

C.1.1 Köra och kompilera

Alla kommandon är tänkt att skrivas i ett terminalfönster.

Simulatorn RACER startas med kommandofilen `runRacer.sh` i katalogen `racer0.5.0/`. Skriv

```
cd racer0.5.0
./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`. För att börja om från utgångspositionen på en bana, utan att avsluta och starta om, tryck `shift` och `R`. För att pausa, tryck `P`.

Regulatorn implementeras i filen `controller.cpp` som ligger i katalogen `racer0.5.0/src`. Använd till exempel `emacs` för att editera. Skriv

```
cd racer0.5.0/src
emacs controller.cpp &
```

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

C.1.2 Logga och bearbeta data

Genom att trycka `L` i RACER startas en loggning. Vid nästa tryck på tangenten sparas en MATLAB-fil `/tmp/logger.mat` med data. Tänk på att denna fil skrivs över vid varje ny loggning. Ladda in datat i MATLAB, använd till exempel `LoadRacerData` (se nedan), och spara det till en fil i er hemkatalog, använd kommandot `save`, för att spara det mer än temporärt.

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

`LoadRacerData` Läser in data från filen `/tmp/logger.mat`.

`ESPplot` Ritar några intressanta storheter i figur 1. 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 drive_x.mat+
```

där `x` är ett index. För att senare göra en jämförande figur mellan säg körning två och fyra, skriv

```
>> load drive_2.mat
>> ESPplot
>> load drive_4.mat
>> ESPplot
```

C.1.3 Programmering

Er regulator ska implementeras i filen `controller.cpp`. Den innehåller en del kommentarer och är delvis självförklarande. 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].

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` eller `AYC_ACTIVE`. När `State` är `AYC_ACTIVE` skrivs det ut ett meddelande på skärmen i RACER.

`float Monitor`
Denna variabel kan användas för att övervaka ett värde i regulatorn. När regulatorn är aktiverad i RACER kommer värdet på variabeln att visas på skärmen.

Exempel

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 = AYC_ACTIVE;`