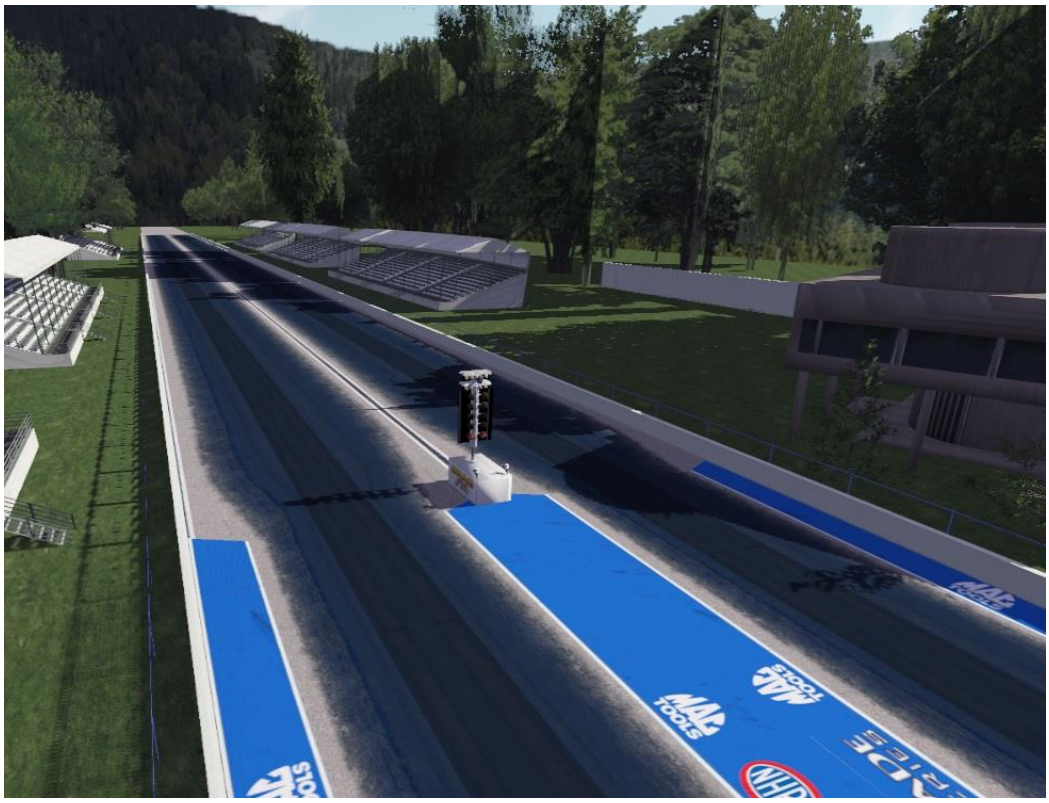
Inlämningsuppgift 1: CASTER dragrace

Växlingssekvens i automatlåda

En automatväxellåda är en uppfinning som gör att föraren av ett fordon slipper växla manuellt, istället väljs den mest lämpliga växeln av ett datorprogram i bilen. Vid konstruktion av en automatväxellåda ställs konstruktören inför otaliga beslut: Vilka utväxlingsförhållanden ska växellådan ha? Hur ska kugghjul, axlar, hylsor, inneslutningar osv dimensioneras för att säkerställa funktionen, och vilket smörjmedel ska användas? Som programmerare kommer du in senare i utvecklingsprocessen, när automatlådans växlingssekvens ska ställas in. Denna process kräver normalt en balans mellan bränsleförbruknings- och prestandakriterier. I denna uppgift programmerar vi dock växlingssekvensen på en Camaro **med det enda målet att köra 402,336m med stillastående start på så kort tid som möjligt**[1]. Växelsekvensen utvärderas i CASTERs simuleringsmotor, anpassad för Matlab. I stället för spelmiljö fås alltså resultat i figur och text.



Figur 1: Dragracebana. Renderat utdrag ur CASTERs simuleringsmiljö.

Mål: Inlämningsuppgiften avser att testa

1) Variabelhantering, loopar, logiska satser
2) Skapa och anropa funktioner
3) Tolka grafer som utdata från program
4) Använda och modifiera förlagor

[1] Sträckan 402,336m, ¼ engelsk mil, är längden på en traditionell dragracingbana.

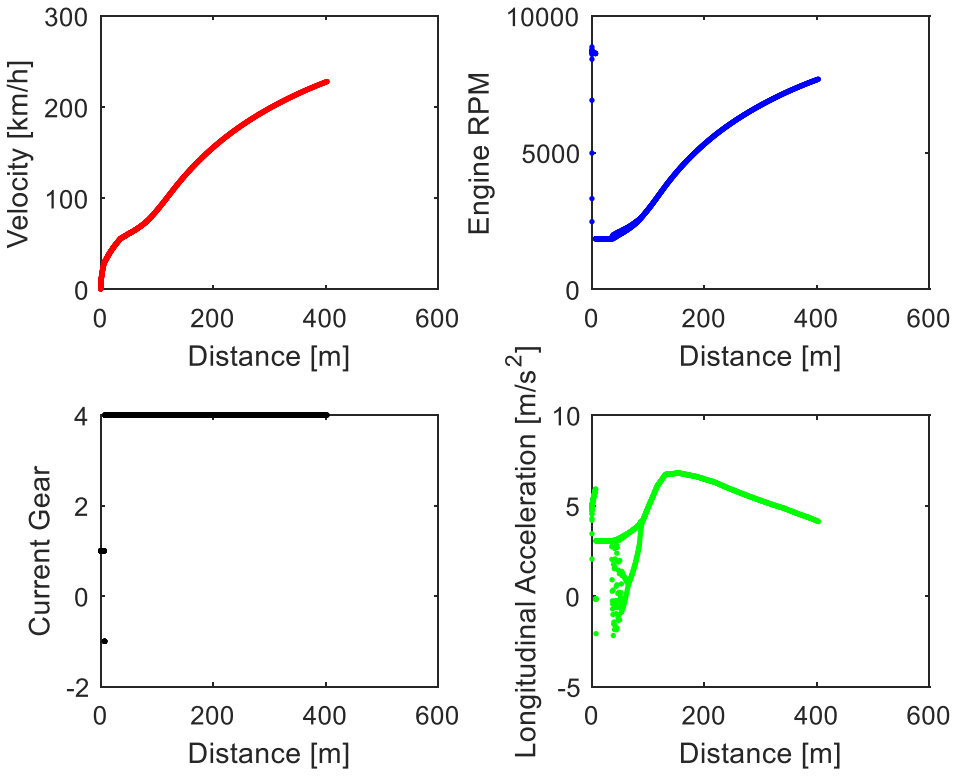
Inlämningsuppgift 1

Att simulera en bils fordonsdynamik sker i en dator genom antagandet att rörelsevariablerna håller sig konstanta under ett mycket kort tidsinkrement. Genom att sedan stega framåt i tiden, inkrement för inkrement, kan en hel rörelse simuleras med stor noggrannhet. För växelsekvensen gäller då att, vid varje tidssteg, utvärdera bilens tillstånd och meddela växellådan vilken växel som är önskvärd för tillfället. Växellådan kommer då, med viss fördröjning, byta till den önskade växeln. **I filen `student_AutomaticGearbox.m` finns en funktion som gör just detta**. I nuläget är den dock mycket ineffektiv; **ditt uppdrag är att förbättra den**.

Figur 2: Tävlingsbilen, en Porsche 935-78 med 845hk. Renderat utdrag ur CASTERs simuleringsmiljö.

Huvudprogrammet (`matlab_Dragster_Assignment`)

I distributionen av filer finns en fil med filändelsen .p – en så kallad *protected file*. Det vill säga en matlabfunktion vars innehåll inte enkelt kan öppnas. Denna fil – `matlab_DragsterAssignment`, innehåller själva simulatorn. När `matlab_DragsterAssignment` exekveras producerar den (efter en liten stunds beräkningar) ett fyrdelat figurfönster som, tillsammans med utskrift till kommandofönstret som innehåller version av koden och själva körtiden, kan användas för att utvärdera växlingssekvensen du har skrivit.



Figur 3: Resultatfigur från Matlabprogrammet `matlab_DragsterAssignment`. Högst upp till vänster plottas momentanhastighet mot körd sträcka, därbredvid motorns varvtal mot körd sträcka. På nedre raden till vänster den växel som används vid varje given punkt längs körsträckan och slutligen till höger accelerationen som funktion av körsträckan.

Utskriften från programmet ser ut som följer:

```
Main Program Version 1.9
InitCommunication Version 1.5
GetTelemetry Version 1.4

Finish time: 25.2541 seconds
ans =
  1×2526 struct array with fields:
    Gear
    Throttle
    RPM
    LongAcc
    Velocity
    Distance
    TimeLap
    TimeTotal
```

Målet med denna inlämningsuppgift är att modifiera växlingssekvensen för att förbättra sluttiden.

Funktionen (`student_AutomaticGearbox`)

Funktionsdefinitionen styrs av simuleringsmiljön (som alltså väntar sig att en funktion med detta namn och anrop ska finnas i den mapp där huvudprogrammet är sparat). Denna ser ut som följer:

```
function [gear_demand, throttle] = student_AutomaticGearbox(Gear, RPM,...
LongAcc, Velocity, Throttle, Distance, TimeLap)
```

Bilen har fyra växlar, och funktionen skickar alltså ut två svarsvariabler:

`gear_demand` – innehåller ett heltal som beskriver vilken växel som önskas [1, 2, 3 eller 4]
`throttle` – hur mycket gas som ska skickas vidare till motorn. Det vill säga, växellådan har möjlighet att justera hur mycket gas som ska ges i varje läge, oavsett vad föraren ger för gas. [ett tal mellan 0 och 1 där 0 = ingen gas, 1 = full gas]

Invariablerna är:

`Gear` – innevarande växel [1, 2, 3 eller 4]
`RPM` – motorvarvtal [rpm]
`LongAcc` – acceleration i körriktningen [m/s^2]
`Velocity` – hastighet i körriktningen [m/s]
`Throttle` – hur mycket föraren gasar [ett tal mellan 0 och 1 där 0 = ingen gas, 1 = full gas]
`Distance` – körd distans [m]
`TimeLap` – tid [s]

I originalutförandet ser koden i funktionen ut på följande sätt:

```
% Convert m/s to km/h
Speed = Velocity * 3.6;
Throttle = 1;
% Check the speed and determine gear
if Speed < 30
    gear_demand = 1;
else
    gear_demand = 4;
end
```

Vilket alltså innebär att automatlådan som används nu startar på första växeln och kör på den till dess hastigheten är 30km/h, då 4:an petas i och sedan används genom hela resten av loppet. Gasen som skickas vidare till motorn lämnas dessutom orörd, det vill säga full gas hela loppet. Det måste naturligtvis gå att förbättra!

Tävling

Målet med inlämningsuppgiften är att effektivisera en automatväxellåda för att kunna uppnå en snabbare körning på en tävlingsbana. En del av utvärderingen av inlämningsuppgiften är att uppnå så stor förbättring som möjligt. De 20 snabbaste koderna kommer att belönas med ett besök hos CASTER i läsvecka 6, där de vinnande studenterna själva får testa sina växlingssekvenser i CASTERs fordonssimulator.[2]

[2] Studenter som inte vinner tävlingen är också välkomna till CASTER, men då utanför skoltid och i mån av tid.

Figur 2: Simulatorn på CASTER sedd från operatörsrummet. Foto: Henrik Sandsjö.

Ledning

Följande avsnitt i kursboken är extra viktiga för just denna inlämningsuppgift:

if-satser: Kapitel 4.1-4.3
funktionsfiler, funktionsanrop: Kapitel 3.7 och 6.1

Formalia

Varje m-fil (skript eller funktionsfil) du skriver i kursen ska ha ett programhuvud med kort förklaring på vad programmet gör och vem som har gjort det, med namn och födelsedag. För funktionsfiler ska in- och utvariabler anges extra tydligt.

Elektronisk inlämning

Inlämningsuppgiften ska rapporteras genom att **`student_AutomaticGearbox.m`, eventuella andra program** du skrivit för att optimera växlingssekvensen och **en rapport om ungefär en A4-sida** som beskriver ditt utvecklingsarbete lämnas in elektroniskt via Canvas senast **onsdagen den 23:e september kl. 23:59**. Ladda upp filerna, **en uppsättning per person**. Programkoden ska vara **väl kommenterad** och **alla filer ska innehålla ett programhuvud** med beskrivning av vad programmet i filen gör. Glöm inte att fylla i **namn** i programhuvudet (så att vi är säkra på vem som gjort uppgiften)!

Rapporten ska innehålla **namn**, **sluttid** och en **kort beskrivning** av hur du arbetat.

Vid misstanke om plagiat skickas ärendet till disciplinnämnden.

Din rapport som bifogas programfilerna vid inlämningen ska beskriva hur du har arbetat för att komma fram till din bättre växlingssekvens. Visa att du arbetat som en ingenjör – dvs metodiskt! **Du kan till exempel basera din kod på teoretiska resonemang om när det borde vara mest optimalt att växla** (se extramaterial i Canvas om du är intresserad av detta spår!) **eller så kan du betrakta det hela som ett rent optimeringsproblem och metodiskt undersöka hur det bäst kan lösas**. Att bara ha testat något och ”råkat” hitta en bättre sekvens är inte tillräckligt! Du behöver ha haft någon form av metodik i ditt testande. Om du känner dig osäker på hur du kan angripa problemet, fråga en övningsledare eller föreläsaren om råd!

Lycka till!