Simulering av luft-bränslereglering med adaptiv parameteruppdatering för ottomotorer

Examensarbete utfört i Fordonssystem
vid Tekniska Högskolan i Linköping
av

Morgan Bergkvist

Reg nr: LiTH-ISY-EX-1721

Simulering av luft-bränslereglering med adaptiv parameteruppdatering för ottomotorer

Examensarbete utfört i Fordonssystem
vid Tekniska Högskolan i Linköping
av

Morgan Bergkvist

Reg nr: LiTH-ISY-EX-1721

Handledare: **Lars Eriksson**
Lars Nielsen

Examinator: **Lars Nielsen**

Linköping, 11 maj 1997.

Abstract

Calibration is a big part during the development of todays engine control-system and is important for improving the engine performance with lower fuel consumption and reduced exhaust emissions. Engine models are used to study different methods with the purpose to systematize calibration of control-system parameters. The engine model used describes a spark ignition engine, and is used to study a control strategy which continuously updates the controller parameters.

Two algorithms for updating parameters in look-up tables, are studied and compared. One of them is chosen and used in a controller to improve air-fuel ratio control (lambda control). Air-fuel ratio control together with the catalyst are important to minimize the exhaust emissions. The main task for the lambda controller is to calculate the amount of fuel to inject in relation to the air inducted into the cylinder. The volumetric efficency describes the amount air drawn into the cylinder, and is therefore an important parameter for air-fuel ratio control. The volumetric efficency depends mainly on the crank-shaft speed and the intake-manifold air pressure and is represented in the model by a two-dimensional look-up table. The controller structure with an on-line update of controller parameters gives promising results in simulation.

Key Word: Lambda control, Engine look-up tables, Engine modelling, Interpolation

Tackord

Jag vill tacka alla på Fordonssystem för en trevlig och lärorik tid. Ett speciellt tack till mina handledare Lars Eriksson och Lars Nielsen för alla idéer och förslag, till Mattias Nyberg för figur 2.1 och till Andrej Perkovic för figur 2.3.

Innehåll

Kapitel 1

Inledning

Dagens motorstyrsystem innehåller många parametrar som måste kalibreras för att motorn skall kunna ge högre prestanda med lägre bränsleförbrukning och minskade avgasutsläpp. Detta ställer allt högre krav på styrsystemens förmåga att kompensera för felaktiga parametervärden som uppstår p.g.a. att motorns åldrande leder till att parametrar förändras samt att styrsystemen endast kalibreras typvis vid produktionen, vilket innebär att styrsystemen inte är optimalt anpassade till varje enskild motor. Genom att införa uppdatering fås möjligheten att korrigera parametrar under det att motorn arbetar, vilket ger bättre reglering samt att det medför enklare kalibrering.

För att undersöka möjligheten att införa uppdatering i styrsystemet studeras här en regulator med uppgift att reglera luft-bränsleförhållandet (lambda) genom att kompensera ut mängden luft som sugs in i cylindern. Det görs genom att uppdatera en tvådimensionell uppslagstabell, även kallad motormapp, som representerar motorns fyllnadsgrad η_{vol}. En simuleringsmodell som bygger på medelvärdesmodellering av ottomotorn används för att testa regulatorn. Lambdaregleringen är en viktig del i motorstyrsystemet för att erhålla minskade avgasutsläpp. Genom att hålla lambda inom ett snävt intervall, det s.k. lambdafönstret, arbetar katalysatorn mest effektivt och därmed erhålls renare avgaser. Två algoritmer för att uppdatera motormappar studeras och jämförs, dessa bygger på linjär interpolering och viktning av hörnpunkter.

1.1 Rapportens uppläggning

Ottomotorn och några av dess viktiga samband beskrivs i kapitel 2. Kapitlet innehåller också ett avsnitt som beskriver motormodelleringen och den simuleringsmodell som använts, samt kvalitativt validerar simuleringsmodellen. Kapitel 3 beskriver två uppdateringsalgoritmer för uppdatering av motormappar som båda bygger på linjär interpolering och viktning av hörnpunkter. Av dessa två väljs en att ingå i lambdaregulatorn, som beskrivs i kapitel 4, för att uppdatera motormappen η_{vol} som representerar motorns fyllnadsgrad. Kapitel 4 beskriver också regulatorstrukturen samt belyser olika problem med lambdaregleringen. Slutligen i kapitel 5 redovisas slutsatser och tänkbara utvidgningar av arbetet.

Implementeringen av de olika funktionerna och modellerna är gjorda i Matlab [6] och Simulink [7]. Simulinkimplementeringen av motormodellen och regulatorerna finns

beskrivna i appendix A, där finns även motormodellens dynamiska ekvationer mer utförligt beskrivna. Appendix B innehåller beskrivningar av de Matlabfunktioner som använts för uppdateringsrutinerna samt lambdaregulatorn.

Kapitel 2

Modellering av ottomotorn

Ottomotorn beskrivs och modelleras i detta kapitel, där första delen beskriver motorn och några av dess viktiga storheter. Motormodellering beskrivs i avsnitt 2.2 och behandlar viktiga dynamiska samband som med medelvärdesmodellering resulterar i en simuleringsmodell bestående av en tillståndsbeskrivning. Motormodellen valideras också kvalitativt.

2.1 Motorbeskrivning

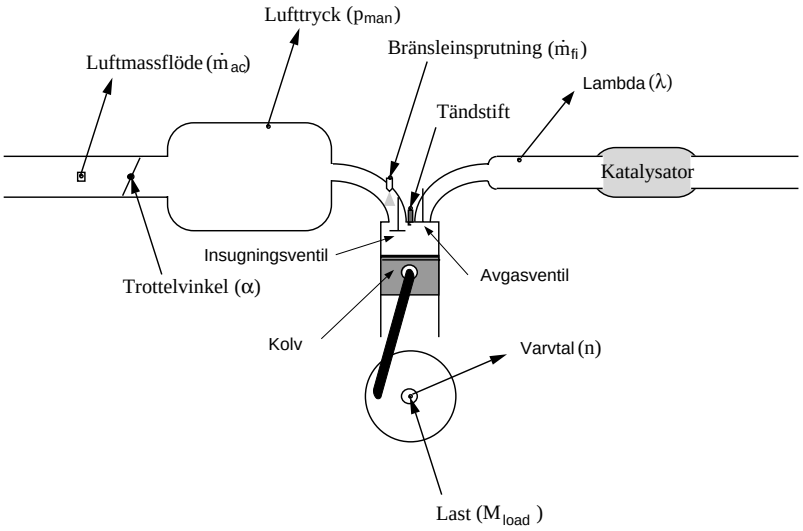
Ottomotorns principiella konstruktion och några av dess viktigaste storheter visas i figur 2.1. Motorn som modellerats är en fyrtaktsmotor vilket innebär att varje cylinders kolv måste utföra fyra takter under en cykel. Motorns vevaxel roterar då två varv eller 720°. De fyra takterna är [2]

Insugningstakt. Startar med kolven i dess övre dödläge (ÖD) och slutar i kolvens nedre dödläge (ND). Avgasventilen är stängd under hela takten. Insugningsventilen öppnas lite innan takten startar och stängs efter dess slut. Under det att kolven dras nedåt sugs ny blandning av luft och bränsle in i cylindern.

Kompressionstakt. Under kompressionstakten är båda ventilerna stängda. När kolven pressas uppåt mot ÖD komprimeras blandningen i cylindern till en bråkdel av dess ursprungliga volym. Mot slutet av kompressionstakten påbörjas förbränningen genom att blandningen antänds av tändstiftet och cylindertrycket ökar hastigt.

Expansionstakt. Det höga gastrycket pressar ned kolven från ÖD till ND och tvingar därmed vevaxeln att rotera. Expansionstakten utvecklar ett arbete som motsvarar ungefär fem gånger det arbete som tillförs i kompressionstakten.

Utblåsningstakt. När avgasventilen öppnas är trycket inledningsvis mycket högre i cylindern än i avgasröret vilket leder till att de förbrända gaserna lämnar cylindern. Kolven pressar också ut resterande gaser ur cylindern då den trycks mot ÖD för att påbörja en ny cykel.



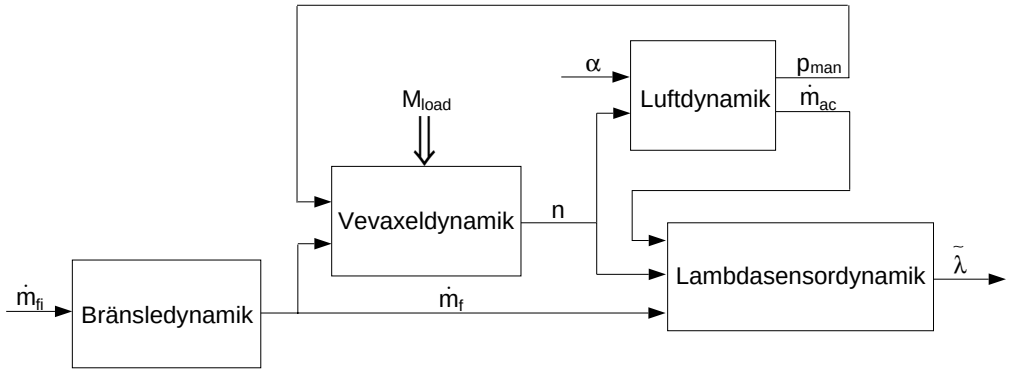
Figur 2.1. *Principskiss av ottomotorn med storheterna $\dot{m}_{ac}$, p_{man}, $\dot{m}_{fi}$, λ, α, n och M_{load}.*

2.2 Motormodellering

I en motormedelvärdesmodell är det i huvudsak två olika typer av förhållande mellan variabler, statiska och dynamiska. Förhållanden som når jämvikt inom några få motorcykler anses vara snabba och beskrivs därför av statiska samband. Förhållanden mellan variabler som tar mellan 10 och 1000 motorcykler för att nå jämvikt beskrivs med dynamiska samband, [1, 4]. Motormodellen som används bygger på tidigare arbeten [3, 10] och är en medelvärdesmodell. Materialet till detta och de följande avsnitten är sammanställt från [1–3, 5, 12].

2.2.1 Modellbeskrivning

Motormodellen består huvudsakligen av delsystemen luftdynamik, bränsledynamik, vevaxeldynamik och lambdasensordynamik. Delsystemens samverkan visas som blockmodell i figur 2.2. Delsystemen med dess dynamik beskrivs i korthet här medans de dynamiska ekvationerna finns utförligare beskrivna i appendix A.



Figur 2.2. *Blockmodell av motorn.*

Vevaxeldynamiken modelleras med Newtons andra lag för roterande massor. Vevaxelns vinkelhastighet n fås genom att ta hänsyn till tröghetsmomentet samt drivande och belastande moment. De belastande momenten är nyttiga lasten M_{load} samt friktionen M_{fric} och det drivande momentet kommer från förbränningen.

En tryckförändring i insugningsröret beror på skillnaden mellan luftflödet in i insugningsröret $\dot{m}_{at}$ och luftflödet in till cylindern $\dot{m}_{ac}$. Luftflödet in i insugningsröret beror på lufttrycket i insugningsröret p_{man} och trottelvinkeln α. När $\alpha = 0^\circ$ är trotteln stängd och då $\alpha = 90^\circ$ är trotteln helt öppen. Luftflödet in i cylindern beror på motorns fyllnadsgrad η_{vol}, vavtalet n och insugningsrörstrycket p_{man}. Fyllnadsgraden beror i sin tur på varvtalet och insugninsrörstrycket, d.v.s. $\eta_{vol}(n, p_{man})$.

Styrsystemet reglerar mängden bränsle som sprutas in i insugningsröret $\dot{m}_{fi}$. En del X fastnar som bränslefilm på väggarna i insugningsröret medans resten $(1 - X)$ går direkt in till cylindern tillsammans med det som förångas av bränlefilmen $\dot{m}_{ff}$. Luft-bränsleförhållandet, A/F (Air/Fuel ratio), definieras som förhållandet mellan luftmassflöde, $\dot{m}_{ac}$, och bränslemassflöde, $\dot{m}_f$ in i cylindern.

$$A/F = \frac{\dot{m}_{ac}}{\dot{m}_f}$$

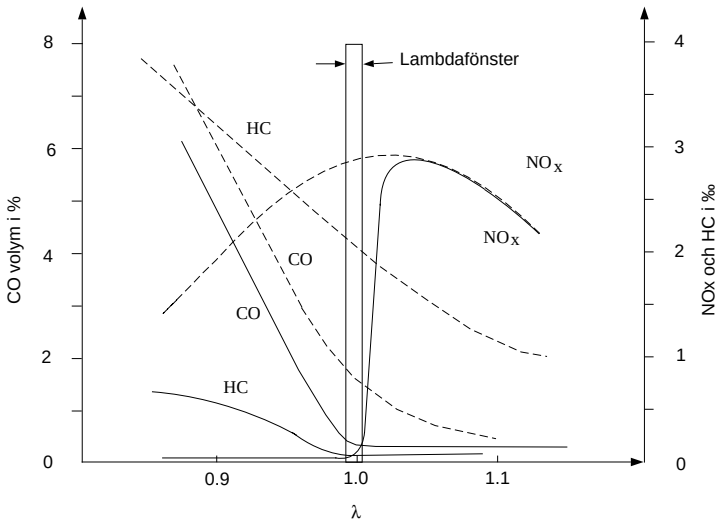
Efter motorn leds avgaserna igenom en trevägskatalysator som minskar utsläppen av koloxid (CO), oförbrända kolväten (HC) och kväveoxider (NO_x) med över 80 %, förutsatt att katalysatorns effektivitet är hög. Katalysatorns effektivitet beror på luftbränsleförhållandet och är högst då luft-bränsleförhållandet är det stökiometriska förhållandet. Det stökiometriska förhållandet beror av bränslets kvalitet men ligger normalt mellan 14.57 och 14.70. Här har värdet 14.67 använts genomgående, vilket innebär att det i vikt räknat behövs 14.67 gånger mer luft än bränsle. Eftersom inte förhållandet mellan luft och bränsle kan mätas direkt används en syremätare som mäter λ, vilket är luft-bränsleförhållandet normaliserat med det stökiometriska värdet.

$$\lambda = \frac{\frac{\dot{m}_a}{\dot{m}_f}}{14.67}$$

För att katalysatorn skall arbeta som effektivast skall λ hållas inom ett snävt område kring 1, lambdafönstret, figur 2.3. När λ ökas så ökar NO_x och då λ minskas ökar CO och HC. Lambdafönstret är det område där NO_x, CO och HC samtidigt har de lägsta värdena.

Lambdasensorn som nästan alla bilar med ottomotor använder sig av är en EGO-sensor (Exhaust Gas Oxygen), vilket är en diskret sensor med omslagspunkt kring $\lambda = 1$ som bara ger information om att blandningen är mager ($\lambda > 1$) eller fet ($\lambda < 1$). En diskret sensor kan därför inte ge någon information om hur mycket bränslemängden skall ökas eller minskas för att erhålla $\lambda = 1$. Den linjära sensorn, UEGO (Universal Exhaust Gas Oxygen), ger det aktuella värdet på lambda och underlättar därmed designen av bra lambdaregulatorer. I motormodellen används den linjära sensorn. En anledning till varför den inte har använts i bilindustrin är att fördelarna inte har motiverat dess högre pris i en produktionsbil. Lambdareglering studeras ytterligare i kapitel 4.

Innan luft-bränsleblandningen kommer till lambdasensorn måste den passera cylindern och förbrännas samt transporteras genom grenröret. Detta resulterar i en tidsfördröjning τ_d som är varvtals- och lastberoende. Lambdasensorn modelleras av ett



Figur 2.3. *Trevägskatalysatorns arbetsområde. Streckade linjer är värdena på CO, NO_x och HC före katalysatorn. Heldragna linjer är värden efter katalysatorn. Katalysatorn ger bäst effektivitet om λ hålls inom lambdafönstret, vilket innebär att CO, NO_x och HC samtidigt har de lägsta värdena.*

första ordningens system. Utsignalen $\tilde{\lambda}$ är beroende av λ och tidsfördröjningen τ_d som modelleras som $\frac{1}{n}$, där n är vevaxelns vinkelhastighet.

2.2.2 Tillståndsbeskrivning

Motormodellen representeras med tillståndsvariablerna,

$$\begin{aligned} x_1 &= n \\ x_2 &= p_{man} \\ x_3 &= \dot{m}_{ff} \\ x_4 &= \tilde{\lambda} \end{aligned}$$

där n är vevaxelns vinkelhastighet, p_{man} är lufttrycket i insugningsröret och $\dot{m}_{ff}$ är bränslefilmens massflöde in i cylindern. Bränslefilmen är det bränsle som fastnar på insugningsrörets väggar. $\tilde{\lambda}$ är utsignalen från lambdasensordynamiken.

Med $\alpha = u_1$, $\dot{m}_{fi} = u_2$ och $M_{load} = d_1$ fås

$$\begin{aligned} \dot{x}_1 &= \frac{60}{2\pi I_{tot}} \cdot \left\{ \frac{\eta_{fc} Q_{HV}}{2\pi\, 60} \cdot \frac{((1-X)\cdot u_2 + x_3)}{x_1} - M_{fric}(x_1, x_2) - d_1 \right\} \\ \dot{x}_2 &= \frac{R\, T_{man}}{V_{man}\, 3600} \cdot \left\{ K_{at} \beta_1(u_1) \beta_2(x_2) + \dot{m}_{at0} - \frac{V_d\, \eta_{vol}(x_1, x_2)}{2\, R\, T_{man}} \cdot 60\, x_1\, x_2 \right\} \\ \dot{x}_3 &= \frac{1}{\tau_{ff}} \cdot (-x_3 + X\, u_2) \\ \dot{x}_4 &= \frac{1}{\tau_\lambda} \cdot \left\{ -x_4 + \frac{1}{14.67} \cdot \frac{V_d\, \eta_{vol}(x_1, x_2)}{2\, R\, T_{man}} \cdot 60\, x_1\, x_2 \cdot \frac{1}{(1-X)\cdot u_2 + x_3} \right\} \end{aligned}$$

Tillståndsvariablerna och ekvationerna för dessa kommer från de olika block som motorn modellerats med, figur 2.2. Modellparametrarna är skattade från uppmätta motordata

i tidigare arbeten [3, 10]. Modellen med dess variabler och konstanter är mer utförligt beskriven i appendix A.1 och i [4].

2.2.3 Modellförändringar

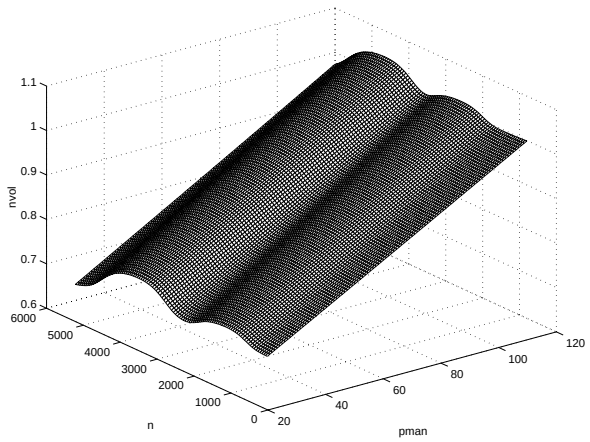
Den stora förändringen i modellen, jämfört med [3], är gjord i modellens luftdynamik där motorns fyllnadsgrad, η_{vol}, har ersatts med en ny funktion, $f(n, p_{man})$. Förändringen är gjord främst för att erhålla ett större arbetsområde. Ekvationen för η_{vol} är

$$\begin{aligned}\eta_{vol} &= f(n, p_{man}) = f_1 + f_2 \cdot n + f_3 \cdot p_{man} + \\ &+ 0.015 \cdot \exp\left(-\frac{\left(\frac{(n-1820-7\cdot p_{man})}{600}\right)^4}{2}\right) + 0.019 \cdot \exp\left(-\frac{\left(\frac{(n-1820-7\cdot p_{man})}{600}\right)^2}{2}\right) + \\ &+ 0.035 \cdot \exp\left(-\frac{\left(\frac{(n-4300-4\cdot p_{man})}{800}\right)^4}{2}\right) + 0.037 \cdot \exp\left(-\frac{\left(\frac{(n-4300-4\cdot p_{man})}{800}\right)^2}{2}\right)\end{aligned}$$

där f_i är

$$\begin{aligned} f_1 &= 0.5669 \\ f_2 &= -7.6071 \cdot 10^{-6} \\ f_3 &= 0.0037 \end{aligned}$$

Funktionen beskriver ett plan med två resonanstoppar vid varvtalen $n = 1820$ och $n = 4300$ som resulterar från stående vågor i insugningsröret, figur 2.4. Planet som beskriver det övergripande uppförandet av η_{vol} är anpassat till mätdata från en verklig motor, där mätdata kommer från [10].



Figur 2.4. *Funktionen beskriver motorns fyllnadsgrad representerad av ett plan med två toppar, där topparna skall motsvara ståendevågfenomen som uppstår genom resonans i insugningsröret.*

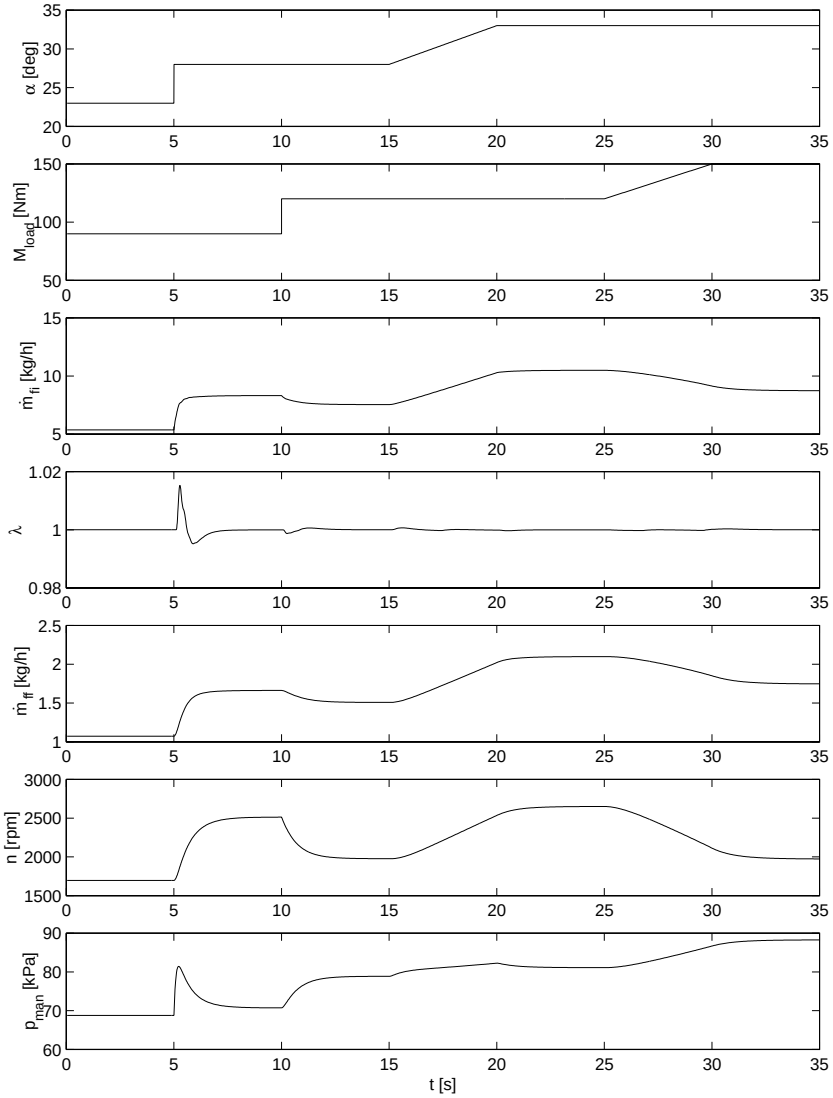
Syftet med motormodellen är att den skall uppföra sig som en verklig motor inom ett begränsat arbetsområde och inte att den skall motsvara en verklig motor exakt. Motormodellens arbetsområde är valt till

$$\begin{aligned} 800 \leq \quad & n \quad & \leq 6000 \\ 30 \leq \quad & p_{man} \quad & \leq 120 \end{aligned}$$

2.2.4 Implementering och verifiering av simuleringsmodell

Motormodellen implementeras i Simulink, appendix A.2, för att användas vid regulatortester. En tidskontinuerlig regulator har tillförts till simuleringsmodellen för reglering av lambda. Regulatorn beskrivs utförligare i kapitel 4.

För att verifiera motormodellen studeras de olika tillståndsvariablerna, styrsignalen $\dot{m}_{fi}$, samt utsignalen lambda, λ, vid simulering med bl.a. steg i trottelvinkel, α, och last, M_{load}, se figur 2.5. Lambda representerar både tillståndet och modellens utsignal.



Figur 2.5. *Modellens uppförande vid simulering av en körning. De intressanta sambanden framträder i tidpunkterna $t = 5s$ och $t = 10s$. Vid $t = 5s$ ges ett steg i alpha vilket leder till att p_{man} ökar kraftigt. En ökning av alpha leder också till att n ökar, vilket medför att p_{man} sedan minskar. Vid $t = 10s$ inträffar ett steg i lasten vilket leder till att n minskar och därmed ökar p_{man}.*

Simuleringen visar att motormodellen uppför sig som en verklig motor, vilket innebär att varvtalet ökar när alpha ökar (gaspådrag) och varvtalet minskar när lasten ökar

(uppförsbacke). Transienterna i lambda beror främst på att luftdynamiken är snabbare än bränsledynamiken, vilket framgår av att jämföra $\dot{m}_{fi}$ och $\dot{m}_{ff}$ med p_{man}. Transienter i lambda behandlas ytterligare i kapitel 4.1.3. Förhållandet mellan p_{man} och insignalen alpha förklaras utförligt i [5], sida 311. Ett viktigt resultat är att transienterna verkligen modelleras och inte döljs.

Kapitel 3

Uppdatering av motormappar

Två uppdateringsalgoritmer för uppdatering av motormappar beskrivs i detta kapitel. Uppdateringsalgoritmerna bygger på linjär interpolering och viktning av hörnpunkter. Den första algoritmen använder likformig viktning och den andra använder avståndsberoende viktning. Av dessa två uppdateringsalgoritmer skall en ingå i en lambdaregulator (kapitel 4) för att uppdatera motormappen som representerar motorns fyllnadsgrad, η_{vol}. Motorns fyllnadsgrad har valts för att den beskriver uppförandet av motorparametrar. För att beräkna mappens värden används en linjär interpoleringsalgoritm som finns beskriven i avsnitt 3.2.

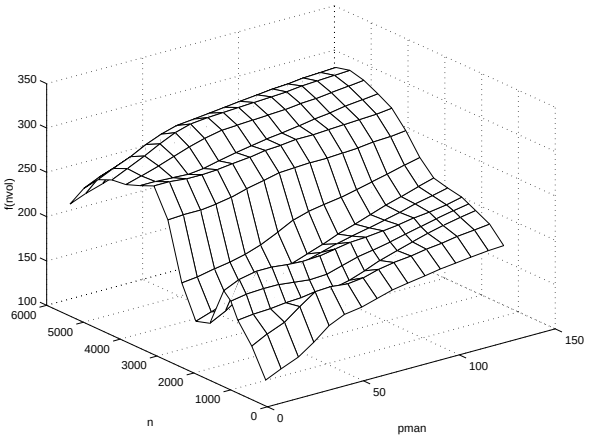
3.1 Motormappar

En motormapp beskriver ett statiskt samband mellan olika parametrar eller variabler och kan representeras som exempelvis $f(x, y)$. Motormappar bestod förr av en knackad plåt som avlästes mekaniskt men har med styrsystemens utveckling blivit ersatt av en mikroprocessor med ett elektroniskt minne som uppslagstabell. Motormapparna, eller tabellerna, innehåller värden på olika motorparametrar och används för att styra motorn genom att ställa ut rätt parametervärde i varje aktuell arbetspunkt. Figur 3.1 visar hur motorns fyllnadsgrad, $\eta_{vol}(n, p_{man})$, kan se ut beroende på varvtalet, n, och lufttrycket i insugningsröret, p_{man}. I figuren är η_{vol} skalad och ett linjärt beroende av p_{man} är borttaget. Funktionen uppvisar också de ståendevågfenomen som tidigare beskrivits.

Införandet av elektroniska datorstyrsystem har inneburit att den tidigare kontinuerliga plåten $f(x, y)$ har diskretiserats i ett antal punkter (x_i, y_i). Motormappen representerar därför bara exakta parametervärden i de diskretiserade arbetspunkterna $f(x_i, y_i)$. För arbetspunkter som befinner sig mellan dessa punkter fås funktionsvärdet genom interpolering.

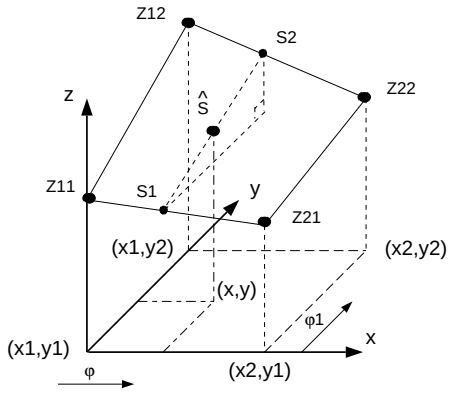
3.2 Interpoleringsalgoritm

Diskretisering av den kontinuerliga ytan innebär att den ej kommer att kunna representeras exakt över hela området eftersom det inte finns tillgång till oändligt många punkter. Antalet punkter har stor betydelse för upplösning, noggrannhet och minnesåtgång.



Figur 3.1. *Exempel på en motormapp som representerar parametern motorns fyllnadsgrad η_{vol} som används i ett befintligt styrsystem. Speciellt kan ståendevågfenomenen observeras som upphöjningar vid varvtal runt 2000 varv per minut.*

Många punkter ger bättre upplösning men ett realistiskt val av matrisens storlek är 16x16 punkter, vilket underlättar implementering i ett styrsystem eftersom färre punkter behöver kalibreras. Punkternas fördelning bör vara sådan att det finns flest punkter där den kontinuerliga ytan har de största oregelbundheterna, detta för att kunna representera det oregelbundna utseendet i mappen. Mappen som används för motorns fyllnadsgrad har inte några extrema oregelbundenheter, figur 2.4, därför används en matris med ekvidistanta punkter som även gör det enkelt att placera ut punkterna. Fyra av matrisens punkter samt området mellan dessa visas i figur 3.2.



Figur 3.2. *En del av matrisen med diskreta punkterna (x_i, y_j) och motsvarande funktionsvärden z_{ij}. Punkten (x, y) representerar den aktuella arbetspunkten med det beräknade funktionsvärdet $\hat{s}$.*

Antag att punkten (x, y) representerar den aktuella arbetspunkten och funktionsvärdet, $\hat{s}$, söks för denna punkt. Förutsatt att punkten (x, y) inte sammanfaller med hörnpunkterna så finns inget exakt funktionsvärde, vilket är det vanligaste fallet. För att erhålla funktionsvärdet $\hat{s}$ tas punkterna s_1 och s_2 till hjälp, som fås genom linjär interpolering mellan punkterna z_{11}, z_{21} och z_{12}, z_{22} i den aktuella punktens x-värde. Funktionsvärdet $\hat{s}$ fås sedan genom linjär interpolering mellan s_1 och s_2 i samma punkts

y-värde. Hjälppunkterna s_1 och s_2 bestämdes i detta fall i x-led men de kan även bestämmas i y-led och båda sätten ger samma värde på $\hat{s}$. Ekvationerna för denna interpolering blir

$$\begin{aligned} s_1 &= z_{11} + \frac{z_{21} - z_{11}}{x_2 - x_1}(x - x_1) = z_{11} + (z_{21} - z_{11}) \cdot \varphi \\ s_2 &= z_{12} + \frac{z_{22} - z_{12}}{x_2 - x_1}(x - x_1) = z_{12} + (z_{22} - z_{12}) \cdot \varphi \\ \hat{s} &= s_1 + \frac{s_2 - s_1}{y_2 - y_1}(y - y_1) = s_1 + (s_2 - s_1) \cdot \varphi_1 = \\ &= (1 - \varphi)(1 - \varphi_1) z_{11} + (1 - \varphi)\varphi_1 z_{12} + \varphi(1 - \varphi_1) z_{21} + \varphi\varphi_1 z_{22} \end{aligned} \tag{3.1}$$

Ekvation (3.1) ger ett uttryck för $\hat{s}$ uttryckt i φ och φ_1 där φ och φ_1 endast kan anta värden mellan 0 och 1. Med

$$\begin{aligned} \alpha_{11} &= (1 - \varphi)(1 - \varphi_1) \\ \alpha_{12} &= (1 - \varphi)\varphi_1 \\ \alpha_{21} &= \varphi(1 - \varphi_1) \\ \alpha_{22} &= \varphi\varphi_1 \end{aligned} \tag{3.2}$$

kan ekvation (3.1) skrivas

$$\hat{s} = \alpha_{11} z_{11} + \alpha_{12} z_{12} + \alpha_{21} z_{21} + \alpha_{22} z_{22} \tag{3.3}$$

Genom att skapa en interpoleringsalgoritm som bygger på linjär interpolering fås på ett enkelt sätt ett approximativt värde på $\hat{s}$. En anledning till att använda linjär interpolering är att det är enkelt och det ger bra resultat. En interpoleringsalgoritm som letar upp de fyra hörnpunkterna och beräknar det interpolerde funktionsvärdet $\hat{s}$ är implementerad som Matlabfunktionen *interp_fcn* och beskrivs i appendix B.

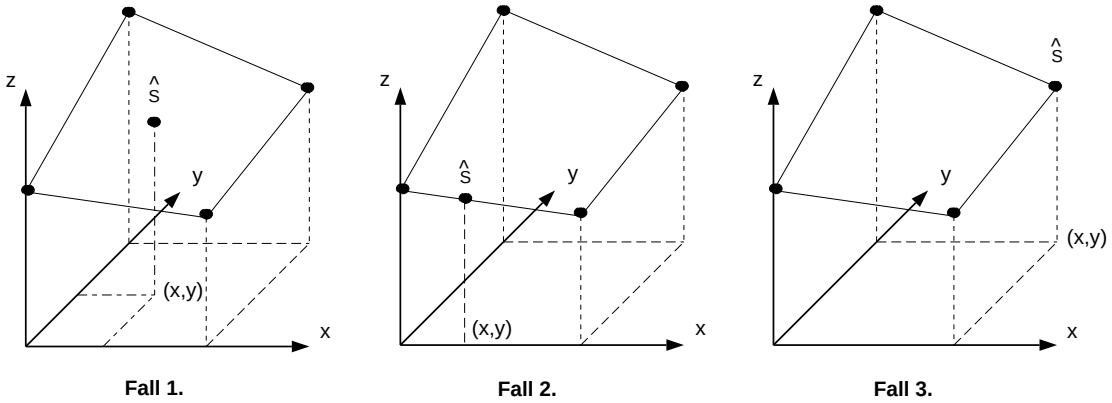
3.3 Uppdateringsalgoritm

Matrisens funktionsvärden $\hat{s}$ kan med interpoleringsrutinens hjälp fås i alla tänkbara arbetspunkter som ligger inom arbetsområdet. Givet ett nytt uppmätt funktionsvärde s i någon arbetspunkt skall också matrisen anpassas till detta värde, d.v.s. några av matrisens punkter skall ändras så att den givna arbetspunktens funktionsvärde kan återskapas så bra som möjligt. Problemet är att bestämma vilka av matrisens punkter som skall ändras och speciellt hur mycket de skall ändras. Två metoder som båda bygger på interpolering och endast tar hänsyn till de närliggande punkterna skall studeras och jämföras.

Vid uppdatering av de närliggande punkterna kan tre olika fall uppstå och de visas i figur 3.3, den aktuella arbetspunkten representeras av punkten (x, y).

Fall 1. Det generella fallet där punkten (x, y) befinner sig mellan alla hörnpunkterna medför att fyra hörnpunkter behöver ändras.

Fall 2. Fallet att punkten (x, y) befiner sig på en av linjerna mellan två hörnpunkter medför att två hörnpunkter behöver ändras.



Figur 3.3. *De olika fallen där punkten (x, y) kan antas vara om hänsyn endast tas till de närliggande punkterna.*

Fall 3. Fallet då punkten (x, y) utgör en av hörnpunkterna medför att en hörnpunkt behöver ändras.

Genom att endast ta hänsyn till de närliggande punkterna kan interpoleringsalgoritmen *interp_fcn* användas för att ta fram de berörda hörnpunkterna. En generell algoritm för uppdatering av punkter i en matris beskrivs enligt

1. Beräkna värdet i den aktuella punkten
2. Beräkna felet i punkten
3. Bestäm hur mycket av felet som skall korrigeras
4. Tag fram de hörnpunkter som måste korrigeras
5. Bestäm hur mycket varje hörnpunkt skall korrigeras
6. Korrigera hörnpunkterna

Det är alltså 6 steg som skall behandlas för att erhålla en uppdateringsalgoritm. Av dessa är det bara stegen 3 och 5 som innehåller nya problem som måste lösas. Steg 3 nämns bara i förbigående i detta kapitel och behandlas mer ingående i kapitel 4. Kärnfrågan är hur mycket varje hörnpunkt skall korrigeras för att uppnå det önskade värdet i arbetspunkten. I fortsättningen antas att punkten (x, y) representerar den aktuella arbetspunkten och Δ är skillnaden mellan det uppmätta funktionsvärdet s och matrisens funktionsvärde $\hat{s}$ i punkten (x, y), d.v.s. $\Delta = |s - \hat{s}|$.

3.3.1 Metod 1: Likformig viktning

Ett enkelt sätt att lösa detta problem är att förändra hörnpunkternas funktionsvärden lika mycket som funktionsvärdet förändras i punkten (x, y). Felet i punkten (x, y) är Δ och korrigeras med Δ_{korr}, vilket ger att de berörda hörnpunkernas funktionvärden skall förändras med Δ_{korr}. Genom att förändra funktionsvärdet i hörnpunkterna lika

mycket bibehålls förhållandet, vilket leder till att rätt funktionsvärde erhålls genom att interpolerings-algoritmen används. Uppdatering av hörnpunkterna z_{ij} blir då,

$$\begin{aligned} \hat{s} &= \alpha_{11}z_{11} + \alpha_{12}z_{12} + \alpha_{21}z_{21} + \alpha_{22}z_{22} && (3.4)\\ \Delta_{korr} &= (s - \hat{s}) \cdot c = \Delta \cdot c && (3.5)\\ \hat{z}_{ij} &= z_{ij} + \Delta_{korr} \quad i,j = 1,2 && (3.6) \end{aligned}$$

där $\hat{z}_{ij}$ är matrispunkternas nya värden. Variabeln c anger hur mycket av felet som skall korrigeras (punkt 3 i den generella uppdateringsalgoritmen). Genom ett lämpligt val av c fås en uppdatering som blir mindre känslig mot störningar eftersom ett fel inte slår igenom helt. Mappen får även en mjukare förändring vid varje uppdatering. En komplett uppdateringsalgoritm som bygger på metod 1 är implementerad som Matlabfunktionen *nvol_up1* och beskrivs i appendix B.

3.3.2 Metod 2: Viktning med avståndsberoende

Ett annat sätt är att förändra hörnpunkternas funktionsvärden genom att vikta hörnpunkterna olika mycket beroende på avståndet till punkten (x, y). Det känns naturligt att de punkter som ligger närmast punkten (x, y) borde vara de som skall bidra mest till en förändring av funktionsvärdet i punkten (x, y). Här används viktningskoefficienterna α_{ij} från ekvation (3.2) som fås från interpoleringsberäkningen av $\hat{s}$. Dessa representerar ett avståndsförhållande mellan punkten (x, y) och respektive hörnpunkt. En fördel är att dessa koefficienter redan är kända från tidigare beräkningar. Uppdateringingen av hörnpunkterna z_{ij} blir då,

$$\begin{aligned} \hat{s} &= \alpha_{11}z_{11} + \alpha_{12}z_{12} + \alpha_{21}z_{21} + \alpha_{22}z_{22} && (3.7)\\ \Delta_{korr} &= (s - \hat{s}) \cdot c = \Delta \cdot c && (3.8)\\ \hat{z}_{ij} &= z_{ij} + \frac{\alpha_{ij} \cdot \Delta_{korr}}{(\alpha_{11}^2 + \alpha_{12}^2 + \alpha_{21}^2 + \alpha_{22}^2)} \quad i,j = 1,2 && (3.9) \end{aligned}$$

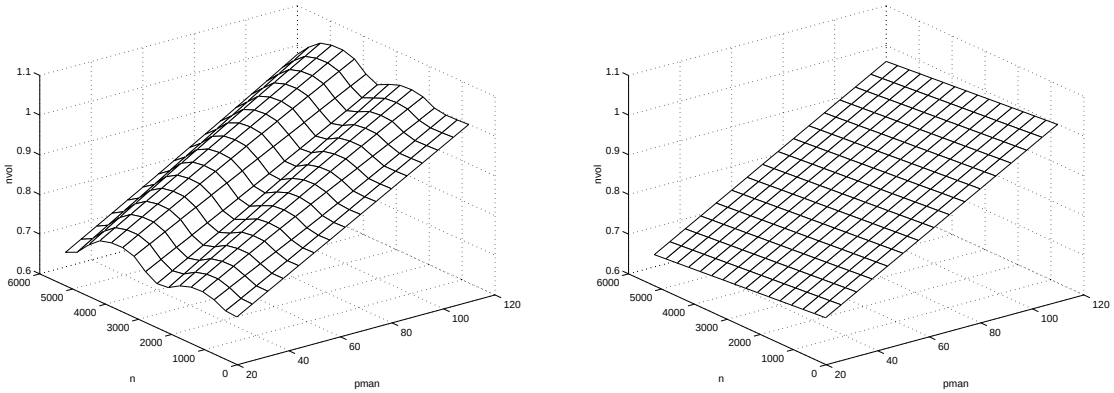
där α_{ij} representeras av ekvation (3.2). Om $\varphi = \varphi_1 = 0.5$ ger metod 2 samma resultat som metod 1. D.v.s. alla hörnpunkterna skall förändras lika mycket. En komplett uppdateringsalgoritm som bygger på metod 2 är implementerad som Matlabfunktionen *nvol_up2* och beskrivs i appendix B.

3.3.3 Andra metoder

En annan tänkbar väg för att lösa problemet är att införa egenfunktioner, där matrisen motsvaras av ett antal egenfunktion. Det finns inget generellt sätt för att ta fram bra funktioner men de kan t.ex. beskrivas av Gaussfunktioner med given varians. En anpassning av funktionen till det aktuella värdet i arbetspunkten innebär en förändring av funktionen i höjdled vilket också leder till att närliggande punkter förändras p.g.a. funktionens varians. Variansen bestämmer även hur många punkter som skall förändras. Detta brukar ibland kallas för självlärande neuronnät.

3.4 Verifiering

För att verifiera och jämföra de båda metoderna behövs en startmatris och en referensmatris. Grunden till dessa matriser utgörs av matrisen i figur 2.4 som representerar motorns fyllnadsgrad. Referensmatrisen *nv16_opt* är en 16x16 matris som är minstakvadratanpassad till η_{vol}, funktionen $f(n, p_{man})$ i avsnitt 2.2.3. Startmatrisen representerar ett plan anpassat till samma funktion. Matriserna visas i figur 3.4. För att få

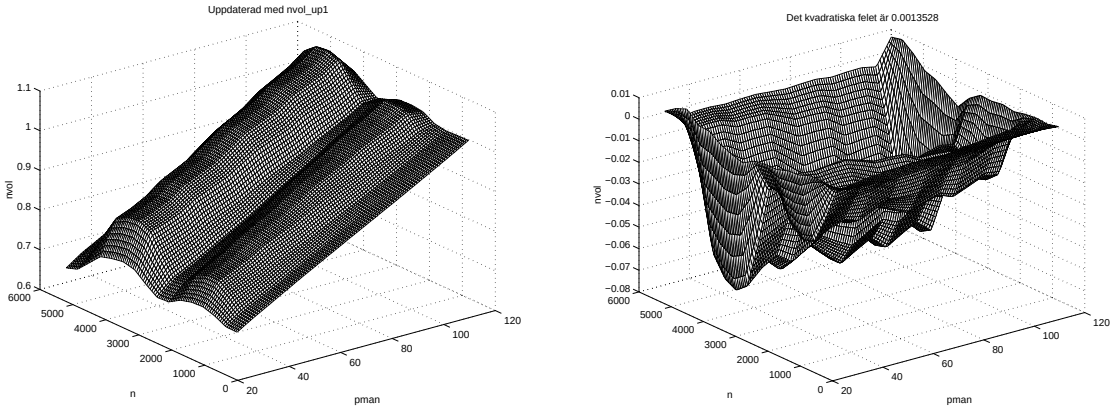


Figur 3.4. *Den vänstra figuren visar referensmatrisen nvol16_opt och den högra visar startmatrisen. Matriserna är representerade i 16x16 punkter.*

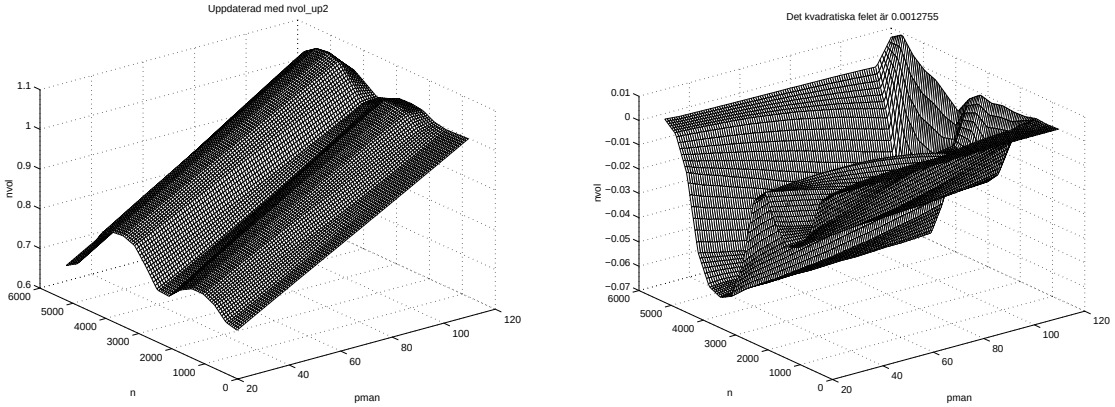
en uppfattning om skillnaden mellan referensmatrisen och de uppdaterade matriserna införs ett kvadratiskt mått, q, som beräknas i 101x101 punkter enligt,

$$q = \frac{\sum_i \sum_j \left(z_{opt}(p_{man}(i), n(j)) - z(p_{man}(i), n(j)) \right)^2}{\sum_i \sum_j 1}$$

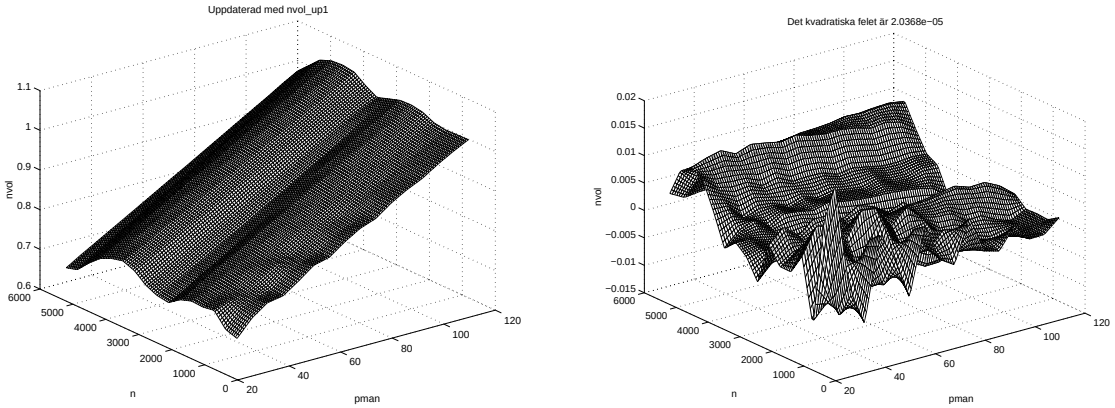
Resultaten efter en uppdatering visas i figurerna 3.5 och 3.6. Resultaten efter femtio uppdateringar visas i figurerna 3.7 och 3.8. Under uppdateringarna har felkorrigeringsfaktorn c valts till 0.25. Av resultaten framgår att uppdateringsalgoritmerna bidrar till att felet minskar. Uppdateringsalgoritmen som bygger på metod 2 ger i båda fallen bättre uppdatering. Felet minskar snabbare och felmatrisen får ett mjukare utseende. För att visa hur funktionerna uppför sig mellan punkterna i matrisen har matriser med 101x101 punkter genererats med linjär interpolation från matriser med 16x16 punkter. Observera att skalorna har förändrats mellan figur 3.5 och figur 3.7 respektive figur 3.6 och 3.8.



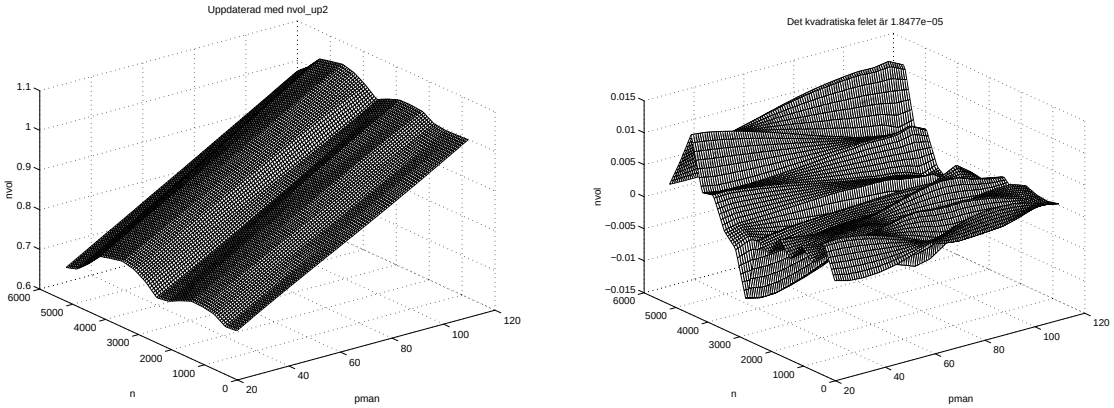
Figur 3.5. *Den vänstra figuren visar η_{vol} efter en uppdatering med nvol_up1 och den högra visar felet. Det kvadratiska felet är 0.0013528.*



Figur 3.6. *Den vänstra figuren visar η_{vol} efter en uppdatering med nvol_up2 och den högra visar felet. Det kvadratiska felet är 0.0012755.*



Figur 3.7. *Den vänstra figuren visar η_{vol} efter femtio uppdateringar med nvol_up1 och den högra visar felet. Det kvadratiska felet är 2.0368e-05.*



Figur 3.8. *Den vänstra figuren visar η_{vol} efter femtio uppdateringar med nvol_up2 och den högra visar felet. Det kvadratiska felet är 1.8477e-05.*

Kapitel 4

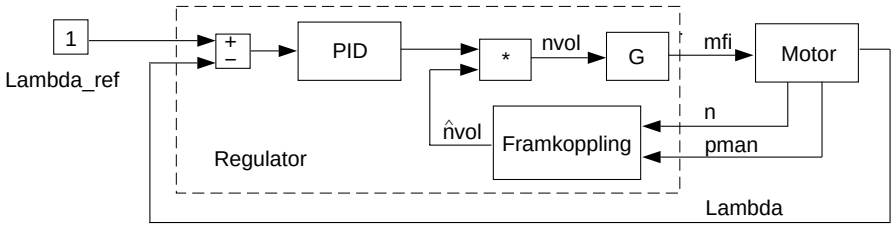
Lambdaregulator

I kapitel 2 diskuterades vikten av bra lambdareglering för att erhålla rena avgaser. En regulator för lambdareglering beskrivs i detta kapitel. Regulatorstrukturen beskrivs i avsnitt 4.1 och innehåller framkoppling och uppdatering av motormappar i syfte att kompensera ut η_{vol} och samtidigt göra en snabbare regulator. Transienters uppkomst och hantering av dessa beskrivs i avsnitt 4.1.3.

4.1 Regulatorstruktur

Den regulator som är vanligast i samband med lambdareglering är PI-regulatorn. PI-regulatorer fungerar oftast bra och de är lätta att implementera. Den PID-regulator som används här ger ett mindre svängigt system men är svårare att implementera på ett bra sätt p.g.a. D-delen.

Vid modellvalideringen i kapitel 2 användes en tidskontinuerlig regulator bestående av en PID-regulator med återkoppling och en framkoppling för prediktering av mängden luft som sugs in i cylindern, $\hat{\eta}_{vol}$. Syftet med regulatorn är att reglera lambda genom att kompensera ut η_{vol}, vilket görs genom att införa en framkoppling. Ett motiv till att prediktera luftmängden i framkopplingen är att regulatorn blir snabbare genom att den anpassas i varje arbetspunkt. Ett blockschema över motor och regulatorstruktur visas i figur 4.1, en utförliggare beskrivning av Simulinkimplementeringen finns i appendix A.



Figur 4.1. *Principskiss över regulatorstrukturen.*

Regulatorn i figur 4.1 är en multiplikativ regulator, vilket innebär att utsignalen från PID-regulatorn multipliceras med det framkopplade värdet i syfte att driva felet till noll,

d.v.s. $\lambda = 1$. Givet ett värde på η_{vol} erhålls styrsignalen $\dot{m}_{fi}$ enligt,

$$\dot{m}_{fi} = \eta_{vol} \cdot \frac{V_d\, 60}{(2\ R\, T_{man}\ L_{st})} \cdot n \cdot p_{man} \tag{4.1}$$

där konstanterna V_d, R, T_{man} och L_{st} redovisas i appendix A.1. I regulatorn antas sambandet (4.1) vara känt för att regulatorn skall kunna reglera bränsleflödet med η_{vol} och inte en skalad η_{vol}. Genom att göra detta antagandet kan η_{vol} jämföras med uppmätta värden på motorns fyllnadsgrad. När η_{vol} är korrekt kompenserad är $\eta_{vol} = \hat{\eta}_{vol}$. Den tidskontinuerliga regulatorn fungerar bra då mappen är känd men den har ingen rutin för uppdatering av motorns η_{vol} mapp.

4.1.1 Tidsdiskretisering av PID-regulatorn

I princip är det svårare att reglera ett system med en tidsdiskret regulator än med en tidskontinuerlig. Detta beror på att en tidsdiskret regulator bara förfogar över en delmängd av de insignaler som en tidskontinuerlig regulator kan använda, nämligen styckvis konstanta signaler. Vid syntes av tidsdiskreta regulatorer måste samplingsintervallet bestämmas, vilket innebär en extra parameter att välja jämfört med tidskontinuerliga regulatorer. Fördelarna med tidsdiskreta regulatorer är att de kan implementeras på en dator och att det är lätt att införa olinjäriteter och villkor av olika slag.

Den tidsdiskreta regulatorn används för att lätt kunna korrigera regulatortillstånd i samband med uppdatering. Den tidskontinuerliga PID-regulatorn som användes för modellvalidering kan skrivas på följande form,

$$u(t) = K\left(e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau - T_D\frac{dy(t)}{dt}\right) \tag{4.2}$$

där u är PID-regulatorutsignalen och e är reglerfelet. Reglerfelet är skillnaden mellan börvärde, $r(t) = \frac{1}{\lambda_{ref}}$, och mätvärde, $y(t) = \frac{1}{\lambda}$. PID-regulatorn har tidsdiskretiserats enligt [8] och har följande utseende

$$\begin{aligned} P_n &= Ke_n \\ I_n &= I_{n-1} + K\tfrac{T_s}{T_I}e_n \\ D_n &= -K\tfrac{T_D}{T_s}\left(y_n - y_{n-1}\right) \\ u_n &= P_n + I_n + D_n \end{aligned} \tag{4.3}$$

där T_s är samplingstiden och K, T_I samt T_D är regulatorparametrarna. Samplingstiden T_s väljs enligt tumregeln att placera 4-8 samplingspunkter på flanken i systemets stegsvar, [11], och regulatorparametrarna K, T_I och T_D trimmas enligt Ziegler-Nichols tumregel för PID-regulatorer, [9].

Systemets stegsvar erhålls genom att göra ett steg i bränsleinsprutningen, $\dot{m}_{fi}$, och därefter beräkna stigtiden i utsignalen lambda. Ett steg i $\dot{m}_{fi}$ med 5% ger stigtiden och motormodellens tidskonstant $\tau_m = 0.12s$. Med 6 samplingspunkter på flanken fås samplingsintervallet $T_s = 0.02s$ som motsvarar samplingsfrekvensen $f_s = 50Hz$.

4.1.2 Framkoppling och uppdatering

Framkoppling används för prediktering av mängden luft som sugs in i motorn och består i huvudsak av en motormapp som motsvarar motorns fyllnadsgrad. Fyllnadsgraden $\eta_{vol}(n, p_{man})$ är en funktion av varvtalet, n, och lufttrycket i insugningsröret, p_{man}, och representeras av en motormapp. Genom att använda interpoleringsalgoritmen *interp_fcn* (tidigare beskriven i avsnitt 3.2) fås värdet i den aktuella arbetspunkten, där arbetspunkten bestäms av n och p_{man}. Motorns fyllnadsgrad förändras då motorn åldras vilket skapar ett behov av att mappen uppdateras. Ytterliggare en anledning till att uppdatering är önskvärt är att motormappar inte anpassas till varje enskild motor utan till varje motortyp.

I regulatorstrukturen, figur 4.1, multipliceras utsignalen från PID-regulatorn, u_n, med det predikterade värdet $\hat{\eta}_{vol}$. När fyllnadsgraden är korrekt skattat, d.v.s. $\hat{\eta}_{vol} = \eta_{vol}$ och systemet svängt in sig är $u_n = 1$. Uppdateringsvillkoret blir att jämföra $u_n - 1$ med en acceptabel felmarginal och om felet är stort uppdateras mappen med funktionen *nvol_up2* (tidigare beskriven i avsnitt 3.3.2).

$$
\begin{aligned}
&if\ (abs\,(u_n - 1) > felmarginal)\\
&\qquad \hat{\eta}_{vol} = nvol\_up2\,(n, p_{man}, \eta_{vol})\\
&\qquad ...\\
&end
\end{aligned}
$$

En uppdatering av motormappen innebär att att $\hat{\eta}_{vol}$ förändras och därmed η_{vol}, eftersom $\eta_{vol} = u_n \cdot \hat{\eta}_{vol}$. För att inte påverka regulatorutsignalen och systemets dynamik vid uppdatering måste regulatortillståndet korrigeras, d.v.s. regulatorns I-del måste förändras så att villkoret

$$\hat{\eta}_{vol} \cdot u_n \quad = \quad \hat{\eta}_{vol\_ny} \cdot u_{n\_ny}$$

uppfylls. Detta är analogt med stötfri övergång vid parameterbyten [8]. Med ekvationen (4.3) för PID-regulatorn fås uttrycket

$$
\begin{aligned}
I_{n\_ny} &= \frac{\hat{\eta}_{vol}}{\hat{\eta}_{vol\_ny}} \cdot u_n - P_n - D_n = \\
&= \frac{\eta_{vol}}{\hat{\eta}_{vol\_ny}} - P_n - D_n
\end{aligned}
$$

där I_{n_ny} är regulatorns nya I-del. η_{vol_ny} fås sedan genom att multiplicera den nya PID-regulatorns utsignal med $\hat{\eta}_{vol\_ny}$.

$$\eta_{vol\_ny} \quad = \quad (P_n + I_{n\_ny} + D_n) \cdot \hat{\eta}_{vol\_ny}$$

Med tidsdiskretisering och införandet av uppdatering beskrivs regulatorstrukturen enligt följande steg,

1. Beräkna PID-regulatorns utsignal
2. Beräkna ett värde på $\hat{\eta}_{vol}$ genom framkoppling
3. Multiplicera utsignalen från PID-regulatorn med $\hat{\eta}_{vol}$

4. Kontrollera om mappen behöver uppdateras
   Om det behövs
   (a) Uppdatera mappen och beräkna ett nytt värde på $\hat{\eta}_{vol}$
   (b) Korrigera regulatortillståndet
   (c) Beräkna en ny PID-regulator
   (d) Multiplicera PID-regulatorns utsignal med $\hat{\eta}_{vol}$
5. Multiplicera produkten med övriga variabler och ställ ut styrsignalen $\dot{m}_{fi}$

Det är viktigt att notera att regulatorutsignalen inte påverkas när motormappen uppdateras.

4.1.3 Transientdetektering

Transienter i lambda orsakas främst av två faktorer. Den första är att bränsledynamiken är långsammare än luftdynamiken och den andra är tidsfördröjningen τ_d (τ_d ingår i ekvation (A.12)) som är tiden från det att luft-bränsleblandningen sugs in i cylindern tills ett värde erhålls på lambdasensorns utgång. Tidsfördröjningen kan delas upp i tre komponenter

1. Tiden genom cylindern
2. Transportfördröjning i gren- och avgasrör
3. Reaktionstiden i lambdasensorn

Transienterna skapar problem vid uppdateringen eftersom de leder till att mappen uppdateras felaktigt och då η_{vol} betecknar ett statiskt samband bör den endast uppdateras under stationära förhållanden. Två åtgärder är vidtagna i regulatorn för att minska inverkan av transienterna. Den första är att använda en prediktering, $\hat{p}_n$, av p_{man} i framkopplingen och i uppdateringen samt i regulatorutsignalen (4.1). Lufttrycket i insugningsröret predikteras under ett sampel enligt

$$\hat{p}_n = p_n + k_{pred}\left(p_n - p_{n-1}\right) \tag{4.4}$$

där k_{pred} är en konstant som trimmats in vid simuleringen. Den andra åtgärden är en enkel transientdetektering tillsammans med lågpassfiltrering av regulatorutsignalen.

$$v_n = k_1 \cdot v_{n-1} + k_2 \cdot w_n \tag{4.5}$$

$$w_n = \frac{u_n - 1}{\left(k_3 \cdot |y_n - y_{n-1}| + 1\right)\left(k_4 \cdot |y_n - 1| + 1\right)} \tag{4.6}$$

Ekvation (4.6) beskriver transientdetektering av regulatorutsignalen. Det viktiga med denna är att w_n är liten vid transienter och vid litet fel, d.v.s. vid en transient kommer skillnaden mellan $y_n = \frac{1}{\lambda_n}$ och $y_{n-1} = \frac{1}{\lambda_{n-1}}$ att öka vilket medför att w_n blir liten om konstanten k_3 trimmats bra. Vid ett fel kommer λ_n att vara skiljd från 1 vilket multiplicerat med konstanten k_4 gör att w_n blir liten. Konstanterna k_3 och k_4 trimmas in vid simulering så att önskat uppförande uppnås. Genom att filtrera w_n genom ett tidsdiskret

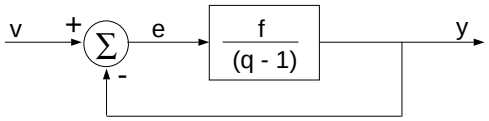
LP-filter, ekvation (4.5), dämpas transienterna ytterliggare vilket gör det naturligt att använda v_n istället för u_n vid uppdatering. Konstanterna k_1 och k_2 fås genom tidsdiskretisering av ett enkelt tidskontinuerligt LP-filter. Eftersom v_n är tillståndet för ett dynamiskt system måste det korrigeras efter uppdatering för att inte dynamiken i systemet skall förändras. Detta görs enligt

$$v_{n\_ny} = v_n \cdot (1 - c) + 1 \tag{4.7}$$

där c är faktorn som anger hur stor del av felet som skall korrigeras.

4.1.4 Val av regulatorparametrar

Genom att anpassa c så att tidskonstanten för uppdateringsalgoritmen blir större än tidskonstanten i motormodellen kan transienter filtreras bort ytterligare. Uppdateringsalgoritmen kan behandlas som ett system bestående av en tidsdiskret integrator med förstärkningen c och tidskonstanten τ_u. Om motormodellens tidskonstant betecknas τ_m blir villkoret att välja c så att $\tau_u > \tau_m$ uppfylls, vilket innebär att motormodellens insvängningsförlopp blir snabbare än uppdateringen. Systemet för uppdatering av en variabel visas i figur 4.2.



Figur 4.2. *Systemet för uppdatering med en tidsdiskret integrator.*

Med tillståndsvariabeln $x = y$ och samplingsintervallet T_s fås systemet

$$\begin{aligned} x(t + T_s) &= (1 - c) \cdot x(t) + c \cdot v(t) \\ y(t) &= 1 \cdot x(t) \end{aligned} \tag{4.8}$$

För att bestämma systemets tidskonstant transformeras systemet i ekvation (4.8) till ett tidskontinuerligt system. På överföringsfunktionsform blir systemet

$$x(t) = \frac{\frac{c \cdot ln(1-c)}{T_s \cdot (1-c) - ln(1-c)}}{(p - \frac{ln(1-c)}{T_s})} \cdot v(t)$$

med tidskonstanten τ_u enligt

$$\tau_u = -\frac{T_s}{ln(1 - c)} \tag{4.9}$$

Ekvation (4.9) ger med villkoret $\tau_u > \tau_m$ och konstanterna $T_s = 0.02s$ och $\tau_m = 0.12s$ villkoret

$$c < 1 - e^{-\frac{T_s}{\tau_m}} = 1 - e^{-\frac{0.02}{0.12}} = 0.154 \tag{4.10}$$

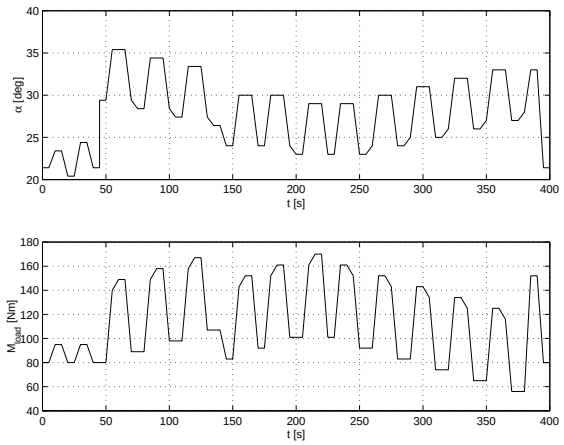
där c väljs till 0.15. Övriga konstanter redovisas i tabell 4.1. En komplett Simulinkimplementering av lambdaregulatorn finns beskriven i appendix A.2.1. Matlabfunktionen *simm_pid* som innehåller regulatorkoden finns beskriven i appendix B.

Variabel	Värde
K	1.695
T_I	0.485/2
T_D	0.485/8
k_{pred}	1.0
k_1	0.9802
k_2	0.0198
k_3	100
k_4	6000

Tabell 4.1. Regulatorkonstanter

4.2 Regulatorverifiering

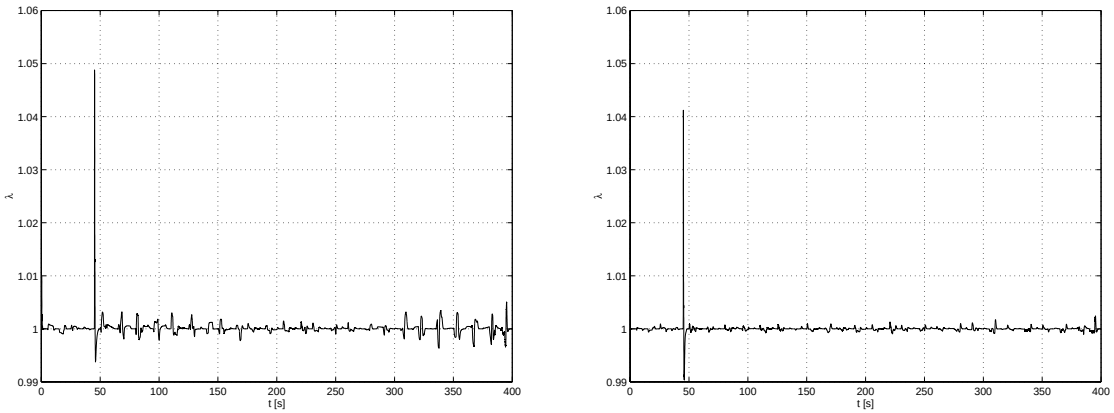
För att verifiera regulatorn har en testcykel för motormodellen tagits fram och använts. Denna skapas genom att förändra trottelvinkeln α och lasten M_{load}, se figur 4.3. Genom att förändra dessa kan motorparametrarna n och p_{man} fås att variera inom det tillåtna arbetsområdet av motormappen. Under körning med testcykeln kommer motor-mappen att anpassas till motorns fyllnadsgrad genom uppdateringen, vilket innebär att regulatorframkopplingen blir bättre efter flera körningar med testcykeln och därmed även utsignalen lambda. När simuleringarna startas representeras regulatormappen för motorns fyllnadsgrad av startmatrisen i kapitel 3.4. Vid simulering av motormodellen och regulatorn studeras endast lambda, λ, som är den intressanta parametern för att verifiera förbättringar.



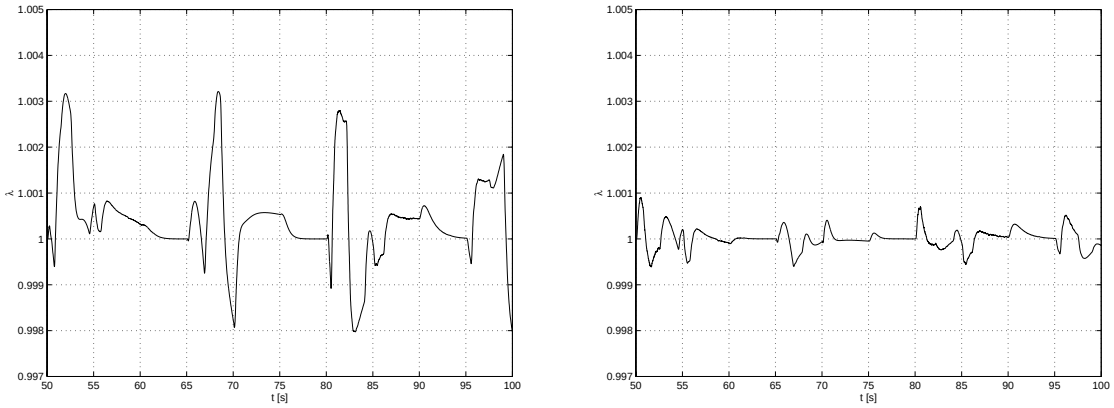
Figur 4.3. *Trottelvinkeln α och lasten M_{load} är valda så att de tillsammans varierar motorparametrarna n och p_{man} inom arbetsområdet. Genom att generera både steg och ramper exiteras systemets dynamik. I de områden där α och M_{load} är konstanta hinner systemet svänga in sig och motorparametrarna ligger still.*

Resultat efter att motormodellen körts med testcykeln visas i figur 4.4. Den vänstra figuren visar λ efter att motormodellen körts en gång med testcykeln och den högra figuren visar λ efter fem gånger med samma testcykel. Av figurerna framgår att felet i lambda har minskat genom att köra testcykeln flera gånger. Det framgår även av

simuleringarna att det finns ett behov av att införa bättre transientkompensering, vilket behövs för att hantera transienten vid $t = 45s$ som inte minskat genom att köra flera gånger. Transienten uppstår genom steget i α vid $t = 45s$ vilket exiterar luftdynamiken som är snabbare än bränsledynamiken. Genom att studera λ under en liten del av cykeln, figur 4.5, framgår tydligare att λ kommer närmare referensvärdet, som är $\lambda = 1$, genom att köra testcykeln flera gånger, d.v.s. uppdatera flera gånger. I figur 4.5 framgår att det efter uppdatering fortfarande är ett fel i λ, detta fel beror på att regulatorn inte kan ta hand om transienterna orsakade av ramper i α. Av simuleringarna framgår att lambdaregleringen förbättras avsevärt och att lambdaregulatorn uppnår önskat resultat.



Figur 4.4. *Den vänstra figuren visar lambda efter en körning med testcykeln och den högra visar lambda efter 5 körningar med testcykeln. Av figurerna framgår att felet minskar genom att köra testcykeln flera gånger. Det som inte minskar är transienternas inverkan, vilket leder till ett behov av bra transientdetektering.*



Figur 4.5. *Figurerna motsvarar delar av ovanstående lambdavärden. Av dessa figurer framgår skillnaden mellan att köra en gång med testcykeln och flera gånger med samma testcykel. Felet har minskat men inte försvunnit help p.g.a. att ramper orsakade av α ger upphov till transienter som inte regulatorn kan kompensera ut. figurer. Av*

Kapitel 5

Slutsatser och utvidgningar

5.1 Slutsatser

En regulator har studerats i syfte att förbättra lamdaregleringen genom att kontinuerligt uppdatera motormapparna för att kompensera ut motorparametern η_{vol}. Regulatorn består av två delar, en PID-regulator och en framkoppling för prediktering av luftmängden som sugs in i cylindern. För att korrigering av regulatortillstånd i samband med uppdatering skulle kunna lösas på ett bra sätt valdes en tidsdiskret regulator. Regulatorn fungerar bra och den uppdaterar den intressanta regulatorparametern. Genom att införa bättre hantering av transienter som slår igenom i lambda skulle regulatorns prestanda kunna förbättras ytterligare.

Två metoder för att uppdatera motormappar har studerats och jämförts. Metoderna bygger på linjär interpolering och viktning av matrispunkter i samband med uppdatering. Den första metoden viktar alla punkter som ingår i uppdateringen lika mycket och den andra använder en viktning som tar hänsyn till avståndet mellan arbetspunkten och övriga punkter. Båda metoderna ger bra uppdateringar men med en avståndsberoende viktning fås en uppdatering som blir bättre i flera avseenden, felet minskar snabbare och uppdateringen blir mjukare.

Motormodellen som används för att verifiera och testa regulatorn är en medelvärdesmodell med fyra tillstånd. Modellens syfte är att beskriva en verklig motors uppförande.

5.2 Utvidgningar

Det finns många tänkbara utvidgningar till detta arbete, några av dem presenteras nedan. Den naturligaste och kanske intressantaste utvidgningen är att använda parameteruppdatering för att prediktera parametrarna X och τ_f i bränsledynamiken. Dessa parametrar bidrar till det s.k. "wall wetting or fuel puddling" problemet, vilket innebär att en del av bränslet fastnar på väggarna i insugningsröret som små pölar som sakta förångas. Detta leder till att motorn får fel bränslemängd under transienter, eftersom pölarna behöver fyllas upp, samt att en del av bränslet sugs in i flytande form. Genom att använda de predikterade värdena för dessa parametrar i en framkoppling för kompensering av bränsledynamiken skulle transienternas inverkan kunna minskas. En

intressant del av problemet är att bestämma adaptiva lagar för hur parametrarna skall uppdateras.

En annan utvidgning är att införa en bättre transientdetektering för att detektera och kompensera för transienter. Den skulle även kunna kombineras med en metod för att prediktera tidsfördröjningen i lambdasensordynamiken.

Ytterligare en intressant utvidgning är att införa självlärande neuronnät för att utföra uppdatering av motormappar.

Referenser

[1] C.F. Aquino. Transient a/f characteristics of the 5 liter central fuel injection engine. *SAE–Technical Paper Series*, (810494), 1981.

[2] BOSCH. *Automotive Electric/Electronic Systems*. Robert Bosch GmbH, 2nd edition, 1995.

[3] Erik Frisk. Model-based fault diagnosis applied to an SI-engine. Master's thesis LiTH-ISY-EX-1679, Vehicular Systems, Linköpings University, 1996.

[4] E. Hendricks. Mean value modelling of spark ignition engines. *SAE–Technical Paper Series*, (900616), 1990.

[5] John B. Heywood. *Internal Combustion Engine Fundamentals*. McGraw-Hill series in mechanical engineering. McGraw-Hill, 1988.

[6] The MathWorks Inc. *Matlab - User's guide*, 1992.

[7] The MathWorks Inc. *Simulink - User's guide*, 1992.

[8] L. Ljung m.fl. Digital Styrning-kursmaterial, 1995. 4 kompendier.

[9] T. Glad och L. Ljung. *Reglerteknik. Grundläggande teori*. Studentlitteratur, Lund, Sweden, second edition, 1989.

[10] M. Nyberg och L. Nielsen. Model based diagnosis for the air intake system of the si-engine. *SAE–Technical Paper Series*, (970209), 1997.

[11] L. Ljung och T. Glad. *Modellbygge och simulering*. Studentlitteratur, Lund, Sweden, first edition, 1991.

[12] Andrej Perkovic and Patrik Berggren. Cylinder individual lambda feedback control in an SI engine. Master's thesis LiTH-ISY-EX-1649, Vehicular Systems, Linköpings University, 1996.

Bilaga A: Simuleringsmodell

A.1 Motormodell

Tabell A.1 redovisar de flesta symboler med enheter som används i modellen.

I_{tot}	Totala tröghetsmomentet $[kg\,m^2]$
n	Vevaxelns vinkelhastighet [rpm]
M_{gross}	Moment utvecklat av motorn [Nm]
M_{fric}	Friktionsmoment [Nm]
M_{load}	Lastmoment [Nm]
η_{fc}	Bränslets omvandlingseffektivitet
Q_{HV}	Bränslets värmevärde[J/kg]
$\dot{m}_f$	Bränslemassflöde in i cylindern [kg/h]
p_{man}	Lufttrycket i insugningsröret [kPa]
p_{amb}	Omgivningens lufttryck [kPa]
T_{man}	Lufttemperaturen i insugningsröret [K]
T_{amb}	Omgivningens lufttemperatur [K]
V_{man}	Insugningsrörets volym $[m^3]$
$\dot{m}_{at}$	Luftmassflöde förbi trottelplattan [kg/h]
$\dot{m}_{ac}$	Luftmassflöde in i cylindern [kg/h]
V_d	Motorns slagvolym $[m^3]$
α	Trottelvinkel [grader]
κ	Specifik värme för luft
τ_f	Tidskonstant för förångningen i bränsledynamiken [s]
X	Andelen av insprutat bränsle som finns i insugningsröret som bränslefilm
$\dot{m}_{fi}$	Insprutat bränslemassflöde [kg/h]
$\dot{m}_{ff}$	Bränslefilmens massflöde [kg/h]
λ	Normaliserat luft-bränsleförhållande
τ_λ	Tidskonstant i lambdasensordynamiken [s]
L_{st}	Stökiometriska värdet för optimal förbränning

Tabell A.1. Symboler och enheter

I modellekvationerna används faktorerna 1/3600 och 1/60. Detta p.g.a. att enheterna för massflöde och vevaxelns vinkelhastighet inte är SI enheter. Massflödesenheten är kg/h istället för m^3/s och vevaxelns vinkelhastighet är rpm istället för rad/s.

Luftdynamik

$$\dot{p}_{man} = \frac{R\,T_{man}}{V_{man}}\frac{1}{3600}\left(\dot{m}_{at} - \dot{m}_{ac}\right) \tag{A.1}$$

$$\dot{m}_{ac} = \frac{V_d\,\eta_{vol}}{2\,R\,T_{man}}\,60n\,p_{man} \tag{A.2}$$

$$\begin{aligned}\eta_{vol} &= \frac{2\,R\,T_{man}\dot{m}_{ac}}{p_{man}\,V_d\,60n} = \\ &= f(n, p_{man})\end{aligned} \tag{A.3}$$

$f\left(n, p_{man}\right)$ i ekvation (A.3) behandlas i avsnitt 2.2.

$$\begin{aligned}\dot{m}_{at} &= f(\alpha, p_{man}) = \\ &= \underbrace{c_t\frac{\pi}{4}D^2\frac{p_{amb}\sqrt{\frac{2\,\kappa}{\kappa-1}}}{\sqrt{R\,T_{amb}}}}_{K_{at}}\beta_1(\alpha)\beta_2\left(p_{man}\right) + \dot{m}_{at0}\end{aligned} \tag{A.4}$$

där β_1 och β_2 ges av

$$\beta_1(\alpha) = 1 - \cos(\alpha - \alpha_0) \tag{A.5}$$

$$\begin{aligned}\beta_2(p_{man}) &= \\ &= \begin{cases} \sqrt{\left(\frac{p_{man}}{p_{amb}}\right)^{\frac{2}{\kappa}} - \left(\frac{p_{man}}{p_{amb}}\right)^{\frac{\kappa+1}{\kappa}}} & \text{, om } \left(\frac{p_{man}}{p_{amb}}\right) \geq \left(\frac{2}{\kappa+1}\right)^{\frac{\kappa}{\kappa-1}} \\ \sqrt{\frac{\kappa-1}{\kappa+1}\left(\frac{2}{\kappa+1}\right)^{\frac{2}{\kappa-1}}} & \text{, annars} \end{cases}\end{aligned} \tag{A.6}$$

Bränsledynamik

$$\ddot{m}_{ff} = \frac{1}{\tau_f}\left(-\dot{m}_{ff} + X\dot{m}_{fi}\right) \tag{A.7}$$

$$\dot{m}_f = (1 - X)\dot{m}_{fi} + \dot{m}_{ff} \tag{A.8}$$

Vevaxeldynamik

$$I_{tot}\frac{2\pi}{60}\dot{n} = M_{gross} - M_{fric} - M_{load} \tag{A.9}$$

$$M_{gross} = \frac{\eta_{fc}\,Q_{HV}\,\dot{m}_f\,\frac{1}{3600}}{n\,2\pi\frac{1}{60}} \tag{A.10}$$

$$M_{fric} = a_0 + a_1 n + a_2 n^2 + (a_3 + a_4 n)p_{man} \tag{A.11}$$

där $M = I\dot{\omega}$.

Lambdasensordynamik

Lambdasensordynamiken modelleras med ett första ordningens system. λ är det verkliga luft-bränsleförhållandet och $\tilde{\lambda}$ är sensordynamikens utsignal. V_{O_2} är lambdasensorns utsignal i volt.

$$\begin{aligned} \dot{\tilde{\lambda}}(t) &= \frac{1}{\tau_\lambda}\left(-\tilde{\lambda}(t) + \lambda(t-\tau_d)\right) = \\ &= \frac{1}{\tau_\lambda}\left(-\tilde{\lambda}(t) + \frac{\frac{\dot{m}_{ac}(t-\tau_d)}{\dot{m}_f(t-\tau_d)}}{14.67}\right) \qquad \text{(A.12)} \\ Vo_2 &= \sigma(\tilde{\lambda}) \qquad \text{(A.13)} \end{aligned}$$

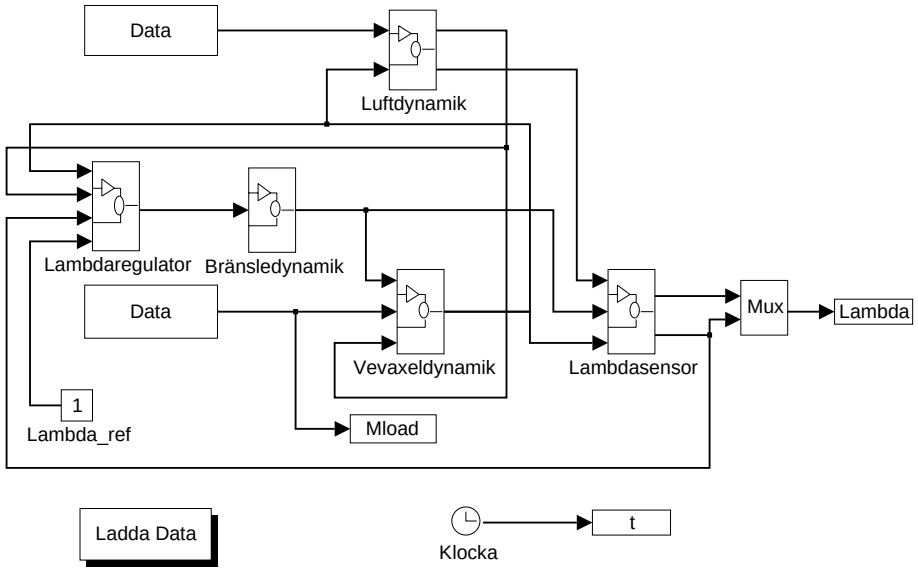
A.1.1 Parametrar

Konstanter som används i modellen redovisas i tabell A.2.

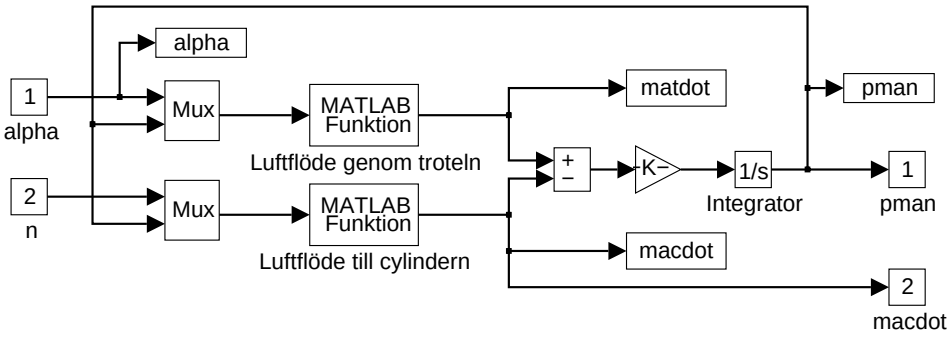
Variabel	Värde
η_{fc}	0.40
τ_f	0.30
τ_λ	0.15
τ_d	195
κ	1.40
X	0.20
R	0.2870
Q_{HV}	$4.3 \cdot 10^7$
$Vman$	0.0065
K_{at}	6109.1
α_0	5.1222
$\dot{m}_{at0}$	6.7557

Tabell A.2. Modellkonstanter

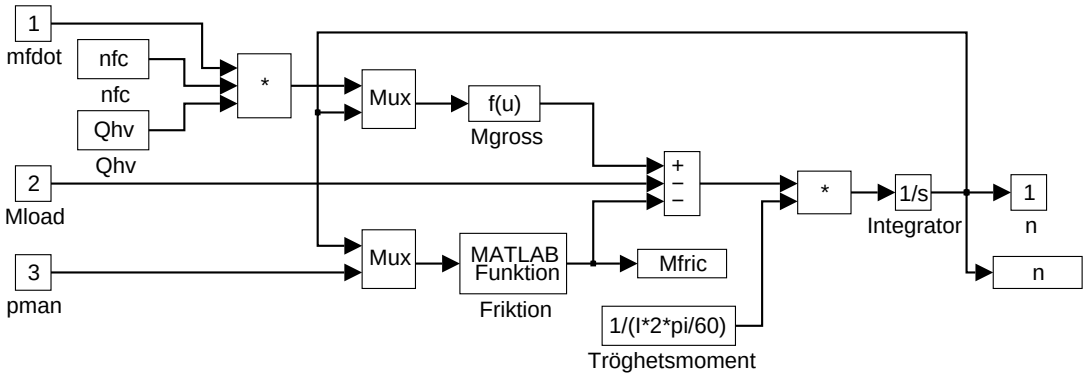
A.2 Implementering av motormodell i Simulink



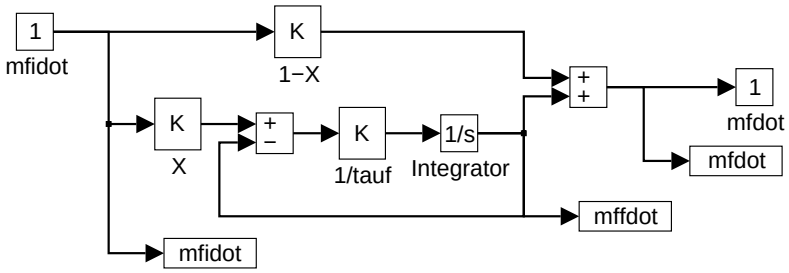
Figur A.1. *Motormodell med regulator*



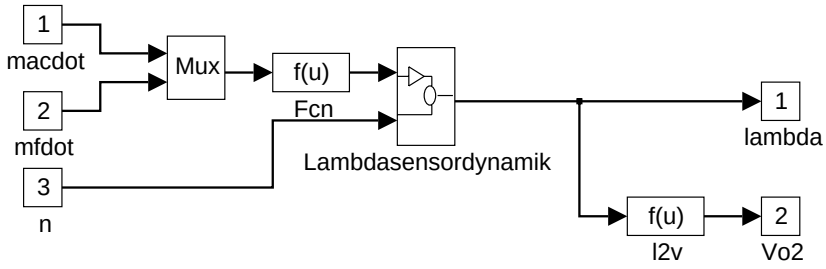
Figur A.2. *Luftdynamik*



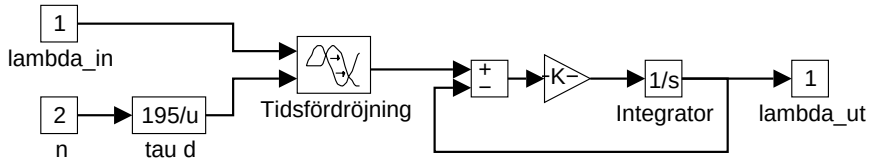
Figur A.3. *Vevaxeldynamik*



Figur A.4. *Bränsledynamik*

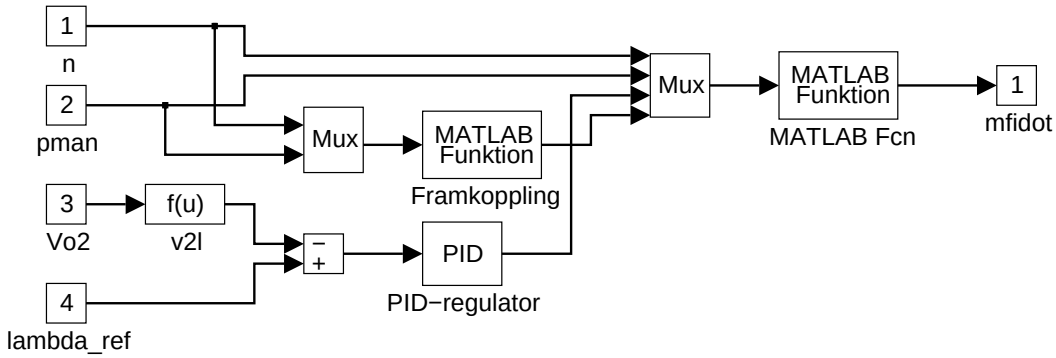


Figur A.5. *Lambdasensor*

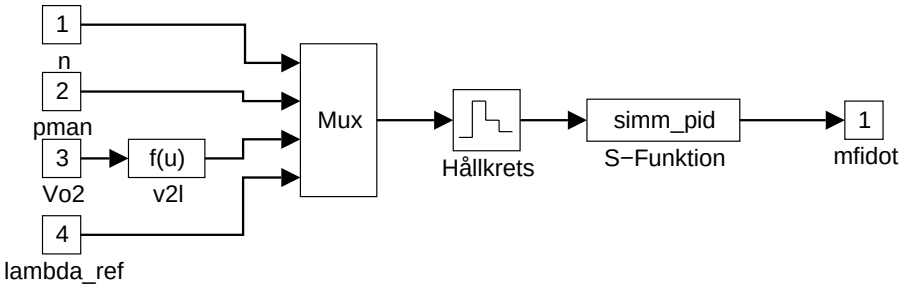


Figur A.6. *Lambdasensordynamik*

A.2.1 Regulatormodeller



Figur A.7. *Tidskontinuerlig lambdaregulator*



Figur A.8. *Tidsdiskret lambdaregulator*

Bilaga B: Matlab funktioner

Funktion interp_fcn

```
function [z2,dot_v,fix_pt,fi,weight]=interp_fcn(x1,y1,z1,x2,y2)

x2_length = length(x2);
y2_length = length(y2);

if nargout > 1 & (x2_length > 1 | y2_length > 1)
  error('The input parameters x2 and y2 must be scalar.')
end

x1_length = length(x1);
y1_length = length(y1);
xl = (x1(x1_length)-x1(1))/(x1_length-1);
yl = (y1(y1_length)-y1(1))/(y1_length-1);

for i = 1:x2_length
  for j = 1:y2_length
    x_norm = 1+(x2(i)-x1(1))/xl;
    y_norm = 1+(y2(j)-y1(1))/yl;

    if x_norm > x1_length | x_norm < 1
      error('Input parameters are out of range.')
    end

    if y_norm > y1_length | y_norm < 1
      error('Input parameters are out of range.')
    end

    x_pt = fix(x_norm);
    y_pt = fix(y_norm);
    fix_pt = [x_pt,y_pt];
    fi(1) = x_norm-x_pt;
    fi(2) = y_norm-y_pt;
    weight = [(1-fi(1))*(1-fi(2));fi(1)*(1-fi(2));
              (1-fi(1))*fi(2);fi(1)*fi(2)];

    if fi(1) == 0 & fi(2) == 0
      dot_v = [z1(x_pt,y_pt);0;0;0];
    elseif fi(1) == 0
      dot_v = [z1(x_pt,y_pt);0;z1(x_pt,y_pt+1);0];
    elseif fi(2) == 0
      dot_v = [z1(x_pt,y_pt);z1(x_pt+1,y_pt);0;0];
    else
```

```
      dot_v = [z1(x_pt,y_pt);z1(x_pt+1,y_pt);
               z1(x_pt,y_pt+1);z1(x_pt+1,y_pt+1)];
    end

    z2(i,j) = weight'*dot_v;
  end
end
```

Funktion nvol_up1

```
function w=nvol_up1(u1,u2,u3)

% Input
% u1:     n        - Crank-shaft speed
% u2:     pman     - Manifold pressure
% u3:     nvol     - Volumetric efficiency

% Output
% w:      nvol_new - The new volumetric efficiency

global n_x p_x nvolum;
c = 0.15;       % c is correction factor.

for i=1:length(n)
  [s,z,f_pt,d] = interp_fcn(n_x,p_x,nvolum,u1(i),u2(i));

  x = (u3(i)-s)*c;

  if x ~= 0
    if d(1) == 0 & d(2) == 0
      nvolum(f_pt(1),f_pt(2)) = z(1)+x;
    elseif d(1) == 0
      nvolum(f_pt(1),f_pt(2)) = z(1)+x;
      nvolum(f_pt(1),f_pt(2)+1) = z(3)+x;
    elseif d(2) == 0
      nvolum(f_pt(1),f_pt(2)) = z(1)+x;
      nvolum(f_pt(1)+1,f_pt(2)) = z(2)+x;
    else
      nvolum(f_pt(1),f_pt(2)) = z(1)+x;
      nvolum(f_pt(1)+1,f_pt(2)) = z(2)+x;
      nvolum(f_pt(1),f_pt(2)+1) = z(3)+x;
      nvolum(f_pt(1)+1,f_pt(2)+1) = z(4)+x;
    end
  end
  w = interp_fcn(n_x,p_x,nvolum,u1(i),u2(i));
end
```

Funktion nvol_up2

```
function w=nvol_up2(u1,u2,u3)

% Input
% u1:     n        - Crank-shaft speed
% u2:     pman     - Manifold pressure
% u3:     nvol     - Volumetric efficiency

% Output
% w:      nvol_new - The new volumetric efficiency

global n_x p_x nvolum;
c = 0.15;         % c is the correction factor.

for i=1:length(n)
  [s,z,f_pt,d,a] = interp_fcn(n_x,p_x,nvolum,u1(i),u2(i));

  x = (u3(i)-s)*c/(a'*a);

  if x ~= 0
    if d(1) == 0 & d(2) == 0
      nvolum(f_pt(1),f_pt(2)) = z(1)+a(1)*x;
    elseif d(1) == 0
      nvolum(f_pt(1),f_pt(2)) = z(1)+a(1)*x;
      nvolum(f_pt(1),f_pt(2)+1) = z(3)+a(3)*x;
    elseif d(2) == 0
      nvolum(f_pt(1),f_pt(2)) = z(1)+a(1)*x;
      nvolum(f_pt(1)+1,f_pt(2)) = z(2)+a(2)*x;
    else
      nvolum(f_pt(1),f_pt(2)) = z(1)+a(1)*x;
      nvolum(f_pt(1)+1,f_pt(2)) = z(2)+a(2)*x;
      nvolum(f_pt(1),f_pt(2)+1) = z(3)+a(3)*x;
      nvolum(f_pt(1)+1,f_pt(2)+1) = z(4)+a(4)*x;
    end
  end
  w = interp_fcn(n_x,p_x,nvolum,u1(i),u2(i));
end
```

Funktion simm_pid

```
function [sys,x0,str,Tsamp]=simm_pid(t,x,u,flag,n_x,p_x,Vd,R,Tman,Lst,Ts)

% Input signals:
```

```
% u(1):     n    - Engine speed
% u(2):     pman - Manifold pressure
% u(3):     Vo2  - The true lambda value
% u(4):     l_ref- Lambda reference
%
% System states:
% x(1):     regI   - Integral part from the controller
% x(2):     oldU   - The old input (1/u(3))
% x(3):     regOut - Output from the controller
% x(4):     oldU   - The old input (u(2))
% x(5):     fout   - Filtered output from PID-controller

global nvolum;

if nargin ~= 11
  if nargin == 0
    flag = 0;
  else
    error('Wrong number of input arguments.');
  end
end

P_del = 0.6*2.825;
I_del = (0.6*2.825)/(0.485/2);
D_del = (0.6*2.825)*(0.485/8);
e_marg = 5e-4;     % An acceptable error in the matrix.

if abs(flag) == 2  % Return discrete states in sys

  err = 1/u(4)-1/u(3);
  regP = P_del*err;
  regI = x(1)+I_del*Ts*err;
  regD = -D_del*(1/u(3)-x(2))/Ts;
  regOut = regP+regI+regD;

  if (regOut < 0)
    regOut = 0;
    disp('Warning negative regulator value');
    regI = -regP-regD;
  end

  sys(5) = 0.9802*x(5)+0.0198*...
           ((regOut-1)/(100*abs(1/u(3)-x(2))+1)/(6000*abs(u(3)-1)+1));

  pPredict = u(2)+1*(u(2)-x(4));
  nvOld = interp_fcn(n_x,p_x,nvolum,u(1),pPredict);
  regOut = regOut*nvOld;
  n_rOld = (sys(5)+1)*nvOld;
```

```
  if (abs(sys(5)) > e_marg)
    nv = nvol_up2(u(1),pPredict,n_rOld);
    sys(5) = sys(5)*0.85+1;
    regI = regOut/nv-regP-regD;
    regOut = (regP+regI+regD)*nv;
  end

  regOut = regOut*(Vd*u(1)*pPredict*60)./(2*R*Tman*Lst);

  sys(1) = regI;
  sys(2) = 1/u(3);
  sys(3) = regOut;
  sys(4) = u(2);

elseif flag == 3  % Return outputs

  sys = x(3);

elseif flag == 4  % Return next sample hit

  % The function shold never enter this block
  error('The sample time depends on the previous block');

elseif flag  == 0  % Initial conditions and size information

  sys=zeros(6,1);
  sys(1) = 0;       % 0 continuous state
  sys(2) = 5;       % 5 discrete state
  sys(3) = 1;       % 1 output
  sys(4) = 4;       % 4 input
  sys(5) = 0;       % 0 roots (unsupported)
  sys(6) = 0;       % 0 direct feedthrough
  sys(7) = 1;       % The system is purely discrete

  Tsamp =[-1 0];

  x0 = [1 1 4.6277 65.365 0];

else

  sys = [];

end
```