Populärvetenskaplig sammanfattning av examensarbetet
Automatisk implementering och analys av Modelica-baserade fixpunkts-regulatorer i Dymola

Ulf Nordström

Bakgrund

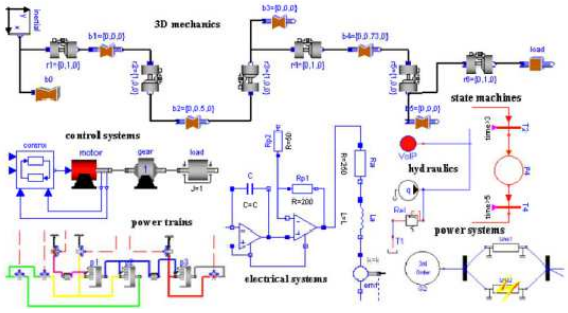
Modellbaserad utveckling är en metodik som blir allt mer vanlig i industrin. Målet är att tidigt i designprocessen använda modellering och simulering för att både specificera och testa processen/produkten. Fördelarna mot att istället använda traditionella metoder och prototyper för testning är främst ekonomiska. Man kan på ett mycket tidigare stadium testa designen och på så sätt validera eller upptäcka konceptuella fel innan dyra prototyper har byggts. Modellbaserad utveckling kan innefatta många typer av scenarion och metoder, och i detta examensarbete har två av dem varit av intresse; Software-In-the-Loop (simulerad mjukvara) och Rapid Prototyping (testning på riktig hårdvara).

Under utvecklingsarbetet används normalt flyttal (en representation av reella tal i datorn) i algoritmer och modeller men i hårdvara (i processorn i den riktiga produkten) får man ofta klara sig med lite enklare heltalsprocessorer utan stöd för flyttal. Detta beror ofta på hårda krav avseende till exempel pris, energiförbrukning, storlek, snabbhet, etc. Heltalsberäkningar är mycket snabbare än motsvarande flyttalsberäkningar då de är enklare att utföra men de har nackdelen att de inte är lika exakta som flyttalsberäkningar.

I det här examensarbetet används fixpunkt för att studera vad som händer i mjukvaran när man räknar lite mindre exakt samt för att automatiskt generera heltalskod som kan köras på heltalsprocessorer för att testa processen.

Modelica och Dymola

Modelica är ett objektorienterat modelleringsspråk som används för att modellera fysikaliska system, exempel i Figur 1. På senare tid har ny utveckling i Modelica gjort att språket även kan användas för styrsystemdelen i ett system.



Figur 1 Olika Modelica-modeller.

Dymola är ett program som används för modellering och simulering av Modelica-modeller. Målet med det här examensarbetet är att använda Dymola för att studera effekter av fixpunkt i Modelica-modeller och generera fixpunktkod för testning på en målplattform.

Fixpunkt

Fixpunkt är ett sätt att representera tal i en dator. I grund och botten kan man säga att det är ett heltal som är skalat med en faktor, i detta sammanhang en multipel av två. Den stora nackdelen med fixpunkt är, som nämnts innan, att man inte kan räkna lika exakt som med flyttal. För att vara mer specifik så är problemet att man inte kan räkna både väldigt noggrant och i ett stort intervall. Man måste hela tiden göra en avvägning om man vill räkna noggrant eller om man vill kunna representera tal i ett stort intervall. Med flyttal har man inte detta problem utan kan räkna väldigt noggrant med tal i ett stort intervall. Att manuellt analysera detta är tidskrävande och ofta svårt, och ett av målen i detta examensarbete är att göra denna analys automatiskt med Dymola för styrsystemdelen av ett system modellerat i Modelica.

Analys

Som ett exempel på en fixpunktsrepresentation ska vi representera talet $y = 1.1$ med 10 bitars noggrannhet (skalning med faktor 2^{10}).

$$y = 1.1 \xrightarrow{fixpunkt} \lfloor 2^{10} \cdot 1.1 \rfloor = 1126 = q$$

där $\lfloor\ \rfloor$ betyder avrundning nedåt. Resultatet, $q = 1126$, är en fixpunktsrepresentation av y med 10 bitars noggrannhet. För att återskapa talet dividerar vi med skalningsfaktorn och får då

$$\tilde{y} = \frac{q}{2^{10}} = q \cdot 2^{-10} = 1.09961 \approx 1.1 = y$$

Som vi kan se ovan så har det införts ett litet fel vid konverteringen till fixpunkt. Felet vid konverteringen beror på noggrannheten i representationen.

Som ett exempel på hur man kan räkna med fixpunktrepresentationer tar vi två tal, $y_1 = 1.1$ och $y_2 = 2.3$, med 10 respektive 8 bitars noggrannhet

$$y_1 = 1.1 \rightarrow \lfloor 2^{10} \cdot 1.1 \rfloor = 1126 = q_1$$

$$y_2 = 2.3 \rightarrow \lfloor 2^{8} \cdot 2.3 \rfloor = 588 = q_2$$

och adderar deras fixpunktsrepresentationer. För att kunna addera q_1 och q_2 måste de ha samma skalning. I exemplet här multipliceras q_2 med 2^2. Vi får då

$$q_1 + q_2 \cdot 2^2 = 1126 + (588 \cdot 4) = 3478$$

För att återskapa resultatet dividerar vi med skalningen och får

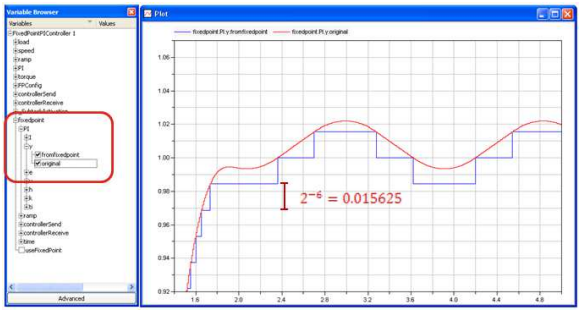
$$3478 \cdot 2^{-10} = 3.396 \approx 3.4 = y_1 + y_2.$$

Här syns åter att ett litet fel introducerats.

Implementering

För att kunna analysera effekten av fixpunkt har utvidgningar gjorts i både Dymola och Modelica. I Modelica-språket har experimentella tillägg, annoteringar, införts för att kunna lägga till information om noggrannhet till variabler. Dymola har utökats med stöd för att analysera och generera kod för fixpunkt. Bl.a. har stöd för intervallanalys, som tillsammans med den experimentella annoteringen kan användas i fixpunktanalysen, introducerats.

Med detta grundläggande stöd kan Software-In-the-Loop scenariot köras för att analysera hur effekterna av fixpunkt påverkar regleringen. För att enkelt kunna jämföra signaler mellan de olika moderna har användargränssnittet för plottning anpassats, se Figur 2 nedan.



Figur 2 Variabelbrowser i Dymola.

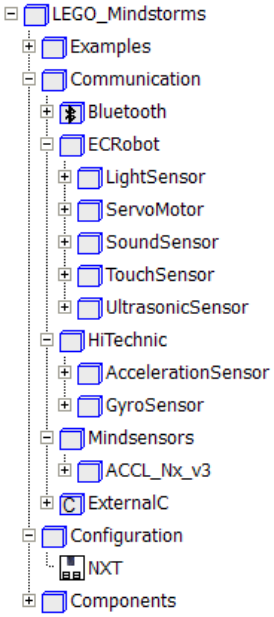
För att kunna testa koden på en plattform i ett Rapid Prototyping scenario har ett interface för en målplattform, Lego Mindstorms NXT (se Figur 3), utvecklats.

Figur 3 Lego Mindstorms tvåhjulig robot.

Interfacet innehåller ett ramverk för att kunna köra den Dymola-genererade koden på Lego-plattformen samt ett Modelica-bibliotek, Figur 4, med komponenter som kan användas för att bygga upp en modell av styrsystemet med sensorer och motorer på Lego-plattformen.

Resultat

Resultatet av examensarbetet har använts i undervisning på LTH, i kursen FRT090 ”Projects in Automatic Control”, åren 2009-2012. Det har också använts i ett annat examensarbete där kod för delar av ett styrsystem till en tvåhjulig balanserande robot med mänsklig förare genererades, Figur 5. Det har också resulterat i två vetenskapliga artiklar som publicerats i konferensmaterialet för Modelicakonferenserna 2006 och 2009.



Figur 4 Modelicabibliotek för Lego Mindstorms.

Figur 5 Tvåhjulig robot.