CHALMERS

Modellering, identifikation och reglering av en quadcopter

Kandidatarbete inom civilingenjörsprogrammen Maskinteknik och Automation och Mekatronik

Andreas Andersson
Christopher Svensson
Daniel Pihlquist
Joakim Larsson
Mattias Wasteby

Institutionen för Signaler och System
Chalmers Tekniska Högskola
Göteborg, Sverige 2014

Kandidatrapport

Sammanfattning

En quadcopter, även kallad drönare, är en typ av flygfarkost som använder sig av fyra motorer för att hovra och flyga. Den kan styras trådlöst av en operatör, eller autonomt med en planerad flygrutt. Quadcoptrar har många olika användningsområden; sök- och räddningsuppdrag, flygfotografi och fritidsflygning. De erbjuder en signifikant fördel över konventionella helikoptrar eftersom de saknar alla komplexa och dyra mekaniska sammankopplingssystem som används i helikoptrar, vilket reducerar priset. Quadcoptrar saknar ett naturligt jämviktsläge, och behöver därför någon form av reglersystem för att stabiliseras.

I detta projektet utvecklades en PDf-regulator baserad på en matematisk modell av quadcoptern. För att verifiera den matematiska modellens noggrannhet utfördes en modellverifikation. Jämförelse av resultat från simulering och det fysiska testet visade att modellen stämde tillräckligt bra för utveckling av ett reglersystem. Eftersom metoden för att utveckla reglersystemet kräver en linjär modell, linjäriserades den. För att kunna implementera det framtagna reglersystemet som mjukvara på en mikrokontroller diskretiserades det med hjälp av MATLAB. På grund av tidsbrist bestod den slutgiltliga implementationen av en annan typ av PID-regulator. Denna regulator var enklare att kalibrera, och kalibrerades experimentellt tills att tillfredsställande stigtid uppnåtts.

Med den experimentellt framtagna regulatorn var quadcoptern stabil nog att flygas av en person utan tidigare erfarenhet. Regulatorn som baserades på den matematiska modellen kunde balansera quadcoptern i en testrigg med tillfredsställande resultat.

Abstract

A quadcopter, also referred to as drone, is a type of aerial vehicle that uses four motors to hoover and fly. It can be controlled remotely by an operator, or fly autonomously with a flight plan. Quadcopters have many different applications; search and rescue missions, aerial photography and recreational flying. They offer one significant advantage over conventional helicopters since they lack all the complex and expensive mechanical linkage used in helicopters, which reduces the price. Quadcopters lack a natural state of equilibrium, and therefore require a control system to stabilize it.

In this project we set out to design a PDf control system based on a mathematical model of the quadcopter. In order to verify the accuracy of the mathematical model, a verification of the model was performed. The comparison of the simulation and the physical test showed that the model was accurate enough to use for the control system design. Since the method for designing the control system requires a linear model of the system, the nonlinear model was linearized. In order to implement the resulting control system on a microcontroller, the system was discretized using MATLAB. Due to time constraints the final implementation involved a different kind of PID controller. This controller offered a more efficient tuning process and was tuned experimentally until a satisfactory rise time was obtained.

With the experimentally developed controller the quadcopter was stable enough to be flown and operated by a person without any previous experience. The controller based on the mathematical model was able to balance the quadcopter in a test rig with good results.

Förord

Följande personer och föreningar har under projektets genomförande bistått med värdeskapande stöd som möjliggjort framsteg i projektet. Vi vill därmed i detta avsnitt tacka de som hjälpt oss nå våra mål.

Vi vill tacka vår handledare Oskar Wigström, doktorand på institutionen för signaler och system på Chalmers, som näst intill varje vecka har träffat och väglett oss genom projektets alla faser. Tack vare denna hjälp har onödigt arbete undvikits och processer har effektiviserats.

Anders Boström, professor på instutitionen för tillämpad mekanik på Chalmers, har bistått med en hjälpande hand vid förståelse av matematiska teorier som tidigare varit obekanta för projektmedlemmarna. Anders har agerat kontaktperson vid utformandet av den matematiska modelleringen, vilket vi vill tacka honom för.

Vid utvecklingen och den fysiska konstruktionen av quadcoptern och dess elektronik har Chalmers Robotförening och Elektrosektionens Teletekniska Avdelning på Chalmers varit till stor hjälp. Vi vill tacka dem för det stöd vi fått i form av lokaler, utrustning och verktyg.

Även prototyplaboratoriet för institutionen tillämpad mekanik och dess ansvarig personal har varit till stor hjälp vid tillverkning av komponenter och testutrustning. Tack till er.

Innehåll

Begrepp och förkortningar

Ad hoc Latin för ”till detta”, egen benämning på en av de regulatorer som implementerades.

Blade Flapping Oscillerande kraft som uppstår när rotorblad färdas i hög hastighet.

BLDC Borstlös DC-motor. En trefasmotor som drivs med en motorkontroller och likströmsmatning.

ESC *Electric Speed Controller*, motorkontroller för att styra BLDC-motorer.

LQR Linjärkvadratisk reglering

I^2C Busskomunikation som används vid inbyggda system.

Inertiala koordinatsystemet Koordinatsystemet relativt jorden.

Mikrokontroller Processorer med in/utportar samt olika periferikretsar.

1

Inledning

En ny typ av flygfordon har tagit världen med storm: quadcoptern. En quadcopter är en helikopter med fyra rotorer och har ett brett användningsområde. Med en filmkamera monterad kan en quadcopter ersätta bemannade övervakningsflygningar och utföra dessa på ett säkrare och mer kostnadseffektivt sätt. Exempelvis vid besiktning av viltskador på åkrar, kontroll av högt belägna högspänningsledningar eller övervakning. Under katastrofer och livräddning kan en quadcopter användas för att leverera proviant, livräddningsutrustning och medicin.

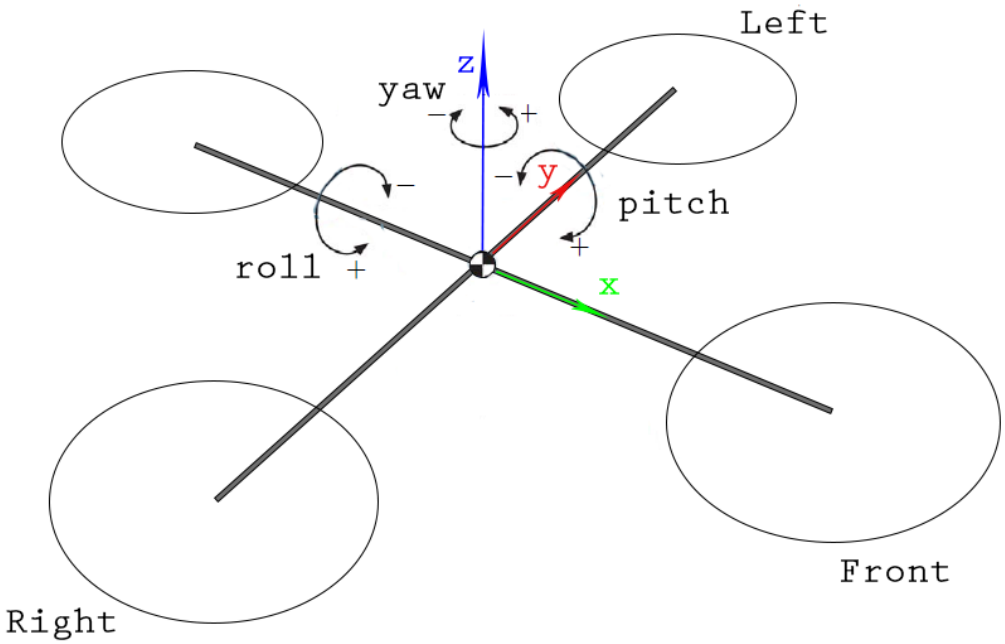
Quadcoptern har en enkel mekanisk konstruktion jämfört med en konventionell helikopter och kan därför verka simpel, det är däremot långt från sanningen. Quadcoptern är ett instabilt system och behöver aktiv reglering för att kunna flygas kontrollerat. Utvecklingen av en quadcopter innefattar därmed signalbehandling, modellering, reglerteknik, simulering, konstruktion, testning och optimering.

Modelleringen beskriver quadcopterns dynamik med matriser och ekvationer. Den användes vid framtagning av reglerystemets parametrar och i kombination med sensorernas övervakning av quadcopterns tillstånd som återkoppling, kan quadcoptern manövreras stabilt.

1.1 Begreppet Quadcopter

Det finns fler benämningar än bara *quadcopter*, termerna *quadrotor* och *quadrocopter* är även vanligt. Jämfört med en helikopter får quadcoptern sin lyftkraft från fyra rotorer istället för en rotor. Vanligast för radiostyrda quadcoptrar är att ha fyra borstlösa likströmsmotorer. Utsignalerna, lyftkraft, *pitch, roll* och *yaw* (se figur 1.1), styrs genom att minska eller öka varvtalet på motorerna.

För att en quadcopter ska kunna flygas kontrollerat krävs ett reglersystem och därmed en mikrokontroller och sensorer. I takt med att utvecklingen av denna teknik har ökat på senare år har priserna för mikroprocessorer och sensorer gått ner. Ett resultat av



Figur 1.1: Yaw, pitch & roll

sjunkande priser på komponenterna är att quadcopters har blivit allt mer förekommande.

En av de första quadcoptrarna var *De Bothezat Helicopter* [1], byggd år 1922. Bredd och längd mättes till 19,8m med en rotordiameter på 8,08m och vikt på 1700kg. Målet var att den skulle klara av att flyga på 100m höjd, men på grund av mekanisk komplexitet, brist på lyftkraft och för hög arbetsbelastning hos föraren, hovrade den som högst på 5m. Quadcoptern var en beställning från *US Army Air Corps*, som önskade en flygmaskin med vertikal lyftförmåga.

Flera tappra försök gjordes de kommande årtiondena men det är inte förrän de senare åren man har kommit fram till något väl fungerande. Med den teknik som finns idag ser man stora möjligheter för att använda flygfordonet inom ett flertal användningsområden.

1.2 Förekomst och användningsområde

Jämfört med en konventionell enrotor-helikopter är en quadcopter både ineffektiv och instabil [2], men har en simpel mekanisk konstruktion som fördel. Nackdelen med en helikopter är komplexiteten i dess mekaniska konstruktion då det krävs att rotorbladen vinklas för att manövrera den. Den mekaniska komplexiteten medför ökad kostnad som undviks med en quadcopter då antalet rörliga delar hålls till ett minimum.

Eftersom quadcopterns materialkostnad är låg är de populära bland RC-entusiaster.

Quadcoptrar testas även för att kunna användas vid hemleveranser [3] och expressleverans av nödpaket i katastrofområden [4].

Det har även blivit populärt bland universitet att utveckla sina egna quadcoptrar för att använda inom olika områden. Forskare och studenter på ETH, Zürich har under de senaste åren använt sig av multipla quadcoptrar som samarbetar för att utföra olika uppgifter. De uppgifter som quadcoptrarna utför kan bland annat vara jonglering med bollar, förflyttning av byggstenar, utföra koreograferade flyguppvisningar med mera.

1.3 Mål

Projektet skall resultera i en quadcopter som med hjälp av det utvecklade reglersystemet kan flygas på ett säkert och stabilt sätt. En person utan tidigare erfarenhet av modellflyg skall med enkla instruktioner kunna flyga quadcoptern.

Quadcoptern och dess reglersystem ska ha utformats efter en utförlig matematisk modell.

1.4 Syfte

Syftet med projektet är att utveckla ett regler- och styrsystem för stabilisering av en quadcopter. Reglersystemet skall tas fram med modellbaserad utveckling, där en matematisk modell används för att beskriva quadcoptern. Den matematiska modellen används för att med analytiska metoder ta fram parametrar till reglersystemet.

Som stöd för utvecklingen skall även olika simuleringsverktyg användas, där den matematiska modellen kan verifieras och jämföras med den fysiska modellen.

För att möjliggöra systematisk utveckling ska även testriggar och programvaror utvecklas för testning och modellverifikation.

2

Konstruktion och System

Kapitlet behandlar den hårdvara samt verktyg som behövs för framtagning av quadcopterns väsentliga delar. Först behandlas de programvaror/verktyg som användes för all programmering och tester, följt av komponentval och systembeskrivning.

2.1 Utvecklingsverktyg

I projektet användes diverse utvecklingsmiljöer för att på ett smidigt vis dela upp lösningen av problemen i mindre delar. För filter- och reglerdesignen inledes arbetet i MATLAB för att testa och ta fram fungerande algoritmer, för att därefter implementeras på quadcopterns mikrokontroller. Om problem uppstodd överfördes arbetet tillbaka till MATLAB.

MATLAB är ett program som används för programmering med fokus på numeriska beräkningar. Det används för att testa algoritmer och utföra simulering av olika system. MATLAB kan även användas för programmering av inbyggda system. Med MATLAB finns många olika applikationer för olika ändamål; till exempel Simulink och SimMechanics.

Simulink är en verktygslåda tillhörande MATLAB och har istället för textgränssnitt ett grafiskt gränssnitt med funktionsblock. Signaler dras med linjer in i olika block, vilket ger en mer överskådlig bild över programmet än vad programkod ger.

SimMechanics är ett verktyg som tillhör Simulink, där mekaniska system med rigida kroppar kan byggas upp och kopplas samman. SimMechanics använder sig av numeriska metoder för att beräkna lösningar för de system som skapats och återskapar och simulerar deras dynamiska egenskaper. SimMechanics använder olika block för att beskriva systemet på samma sätt som Simulink. Blocken symboliserar kroppar och leder, där lederna tillåter kropparna att ha en eller flera frihetsgrader. MATLAB användes till många delar, men ej för realtidsövervakning av systemet, då användes Processing.

Processing är en utvecklingsmiljö för Java framtagen av Ben Fry och Casey Re-

as från MIT [5]. Processing gör det möjligt att snabbt skriva Java-applikationer med standardbibliotek och många tilläggsbibliotek.

För programmering av inbyggda system kan MATLAB användas men under projektet har hårdvaruutvecklarnas egna programvaror använts, Arduino IDE och MPLAB. Arduino IDE är en utvecklingsmiljö för C++ där programvara för Arduino författas. Arduino IDE är baserat på Processing. MPLAB är en kompilator tillhörande hårdvara från Microchip. Det är en nödvändig programvara för programmering av PIC-processorn som benämns i rapporten.

2.2 Sensorer

För att ett system skall kunna ha ett återkopplat reglersystem, måste systemets tillstånd mätas. Tillstånden mäts med sensorer där valet av sensorer beror på vilken storhet som önskas att regleras. I quadcopterns fall återkopplas vinkel och vinkelhastighet. Lämpligt val av sensorer kan vara accelerometer och gyroskop. Systemet kompletteras ibland med en magnetometer. Önskas även reglering av höjd räcker ej dessa sensorer, utan barometer eller ultraljudssensor kan anses lämplig. Barometern mäter lufttrycket som ändras beroende på höjd, och ultraljudssensorn mäter tiden det tar för en utsänd ljudpuls att färdas till marken och reflekteras tillbaka till sensorn.

En accelerometer mäter den acceleration som verkar på sensorn [6]. Det inkluderar både linjära rörelser av sensorn och dess orientation relativt gravitationsvektorn. För att avgöra vinkeln relativt gravitationsvektorn mäts tyngdaccelerationen och vinkel beräknas enligt, där accelerationen mäts i x, y och z som visas i figur 1.1,

$$\phi = \arctan \frac{\ddot{x}}{\sqrt{\ddot{y}^2 + \ddot{z}^2}}, \tag{2.1}$$

$$\theta = \arctan \frac{\ddot{y}}{\sqrt{\ddot{x}^2 + \ddot{z}^2}} \tag{2.2}$$

och mätvärden från de rörelser som sensorn utsätts för kan ses som störningar. Bidraget från tyngdaccelerationen till mätdata förväntas vara lågfrekvent medan bidraget från rörelse till mätdata förväntas vara högfrekvent.

Ett gyroskop mäter med den vinkelhastighet sensorn utsätts för [7]. Efter integrering kan den relativa vinkeln utläsas. Då endast relativ vinkel beräknas antas det skattade värdet avvika från verkligheten efter en viss tid. Eftersom absolut vinkel inte kan beräknas med gyroskop anses de högfrekventa förändringarna av mätdata vara av intresse medan de lågfrekventa anses vara störningar.

Med en magnetometer kan absolut vinkel beräknas utifrån sensorns orientation relativt det magnetfält som omger sensorn. Det magnetfält som önskas mätas är jordens magnetfält, då det ger en absolut referens. Störningar kan dock uppkomma från omkringliggande elektronik. Därav gäller samma för magnetometern, som för accelerometern, att lågfrekventa förändringar av mätdatan är relevant och högfrekventa förändringar anses som störningar.

Avläsning från sensorerna utfördes inledningsvis med en *Arduino Uno*, Atmega328 mikroprocessor, och I^2C-kommunikation. Beräkningar för vinkel, filter och motorsignaler utfördes med samma processor.

När accelerometern ligger plant, normalvektorn parallell med gravitationsvektorn, utsätts den för 1g i z-led och 0g i x- och y-led enligt figur 1.1. Datan från accelerometern är dock ej uttryckt i förhållande till gravitationen utan behöver skalas. Skalfaktorn ges av sensorns datablad och en statisk avvikelse benämnd bias behöver subtraheras för kalibrering. Vinkeln för pitch respektive roll beräknas med

$$\hat{\theta} = \arctan \frac{\ddot{x}_m - \ddot{x}_{bias}}{\sqrt{(\ddot{y}_m - \ddot{y}_{bias})^2 + (\ddot{z}_m - \ddot{z}_{bias})^2}} \tag{2.3}$$

$$\hat{\phi} = \arctan \frac{\ddot{y}_m - \ddot{y}_{bias}}{\sqrt{(\ddot{x}_m - \ddot{x}_{bias})^2 + (\ddot{z}_m - \ddot{z}_{bias})^2}}. \tag{2.4}$$

Om monteringen av accelerometern ej är helt parallell med quadcoptern kan en statisk avvikelse ske benämnd *bias*. Den beräknas i en kalibreringsfas där quadcoptern ställs plant och kan sedan subtraheras i resterande mätningar.

Vinkelestimering för gyroskopet görs genom integrering enligt, där h är samplingstiden,

$$\hat{\theta}_k = \hat{\theta}_{k-1} + (\dot{\theta}_{k_m} - \dot{\theta}_{bias}) \cdot gyro_{scale} \cdot h \tag{2.5}$$

där skalfaktorn ges av databladet. Vinkeln för alla tre axlar beräknas med ekvation 2.5, men med andra mätvärden.

2.3 Filter

Sensorer är aldrig ideala utan producerar alltid en viss mängd brus, vilket gör att mätningarna måste filtreras. Bruset kan bestå av naturligt mätbrus från sensorn, eller dynamiska effekter som ej önskas mätas, exempelvis linjär acceleration från accelerometern.

Komplimentärfiltret är, jämfört med Kalmanfiltret som förklaras mer ingående senare, lättare att förstå och implementera och kräver mindre processorkraft. Den använder sig av mätdata från två oberoende sensorer och gör en viktad sammanvägning av dem för att skapa en skattning av det nuvarande tillståndet. Komplementärfiltret kan då ses som ett kombinerat låg- och högpassfilter [8]. Den ena sensorn har ett frekvensinnehåll där endast lågfrekventa signaler är intressanta och högfrekventa signaler anses vara mätbrus eller störningar, och omvänt för den andra sensorn.

Dock kommer komplementärfiltrets simplicitet med nackdelar; på grund av att all systemdynamik inte tas med i beräkningarna skapas osäkerheter. Det finns även begränsningar i filtrets transientsvar. De enda systemegenskaper som tas med är uppskattningar av hur snabb systemets dynamik är, som används till att bestämma filtrets tidskonstant.

Om $\alpha_1(t)$ och $\alpha_2(t)$ är två observationer av tillståndet $\alpha(t)$ så kan ett sammanvägt estimat beskrivas med

$$\hat{\alpha}(t) = a \cdot \alpha_1(t) + (1 - a) \cdot \alpha_2(t) \tag{2.6}$$

där $\alpha_1(t)$ är den komponent med högfrekvent signalbrus och $\alpha_2(t)$ är den med lågfrekvent signalinnehåll.

Konstanten a beräknas med ekvation (2.7) där h är systemets samplingstid och τ är låg- och högpassfiltrets tidskonstant.

$$a = \frac{\tau}{\tau + h} \tag{2.7}$$

För att endast inkludera de bra delarna från accelerometern respektive gyroskopet och kombinera det till en tillförlitlig signal används ekvation (2.6) för gyroskopet och accelerometern enligt följande,

$$\hat{\theta}_k = a \cdot (\hat{\theta}_{k-1} + \dot{\theta}_{k_{gyr,m}} \cdot h) + (1 - a) \cdot \hat{\theta}_{k_{acc,m}} \tag{2.8}$$

där h är samplingstiden. Roll kan beräknas fram på samma sätt. Men då yaw ej kan beräknas med accelerometern används endast mätdata från gyroskopet. Om gyrot visar sig drifta för mycket kan mätdata från magnetometern användas i samma syfte som accelerometern används för pitch och roll.

För att ta fram ett komplementärfilter som fungerar, och bekräfta dess kapacitet, samplades mätvärden in i MATLAB. I MATLAB testas alla beräkningar och funktioner för att sedan överföras till Arduinon med C++ och slutligen till PIC32 med C.

Kalmanfiltret [9] är en annan typ av filter som sammanväger en skattning baserad på en linjär modell av systemet och en uppmätt skattning av systemets nuvarande tillstånd.

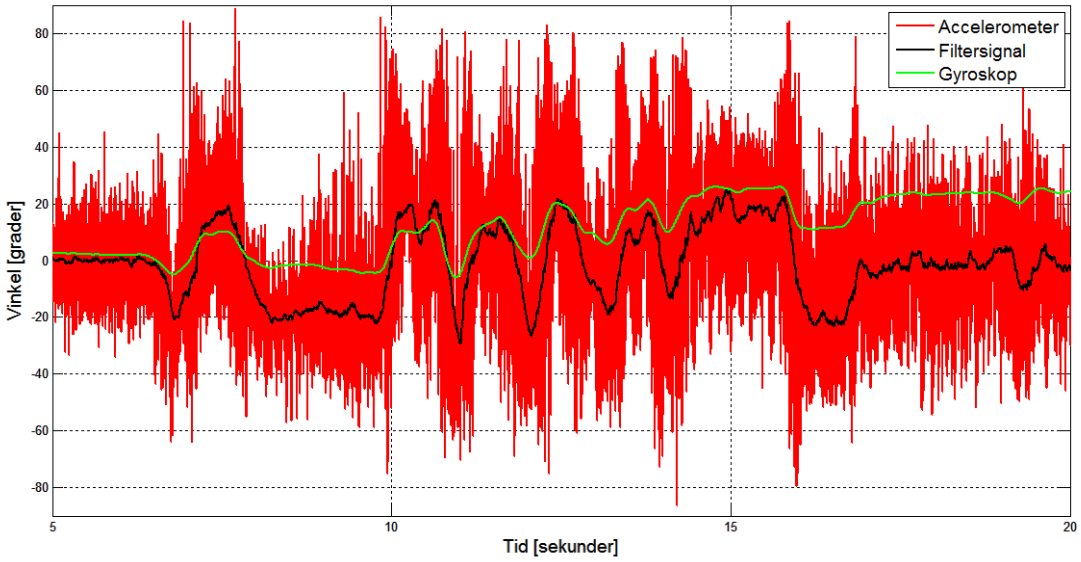
En av kalmanfiltrets fördelar är att det tillåter en att använda en ofullständig modell av det verkliga systemet. Kalmanfiltrets största nackdel är dess komplexitet [10]; det är svårt för någon som inte tidigare arbetat med det eller lärt sig grundläggande teori för det att göra en implementation av det. Jämfört med komplementärfiltret sker det fler beräkningar i varje tidsteg, vilket resulterar i att processorn belastas mer.

2.4 Mikrokontroller

Trots att positiva resultat uppnåddes med ursprungliga processorn, Atmega328, ansågs den inte ha tillräckligt bra prestanda för att nå projektmålen. Den hade dessutom ingen trådlös styrning, vilket gjorde att tester i luften försvårades signifikant. Därför inleddes arbete med en annan processorarkikektur, PIC32. PIC32 är en 32-bitars processor med en klockfrekvens på 80Mhz, som i jämförelse med Atmega328 8-bitars processor och 16MHz möjliggör en snabbare uppdatering av motorerna. Tillsammans med hårdvaran på PIC32 fanns även en blåtandsmodul som förenklade arbetet, då data kunde skickas till och från processorn trådlöst. Blåtandsmodulen möjliggjorde realtidsplottning av sensordata samt justering av parametrar utan att styrkortet behövde vara inkopplat med kablar.

Resultat

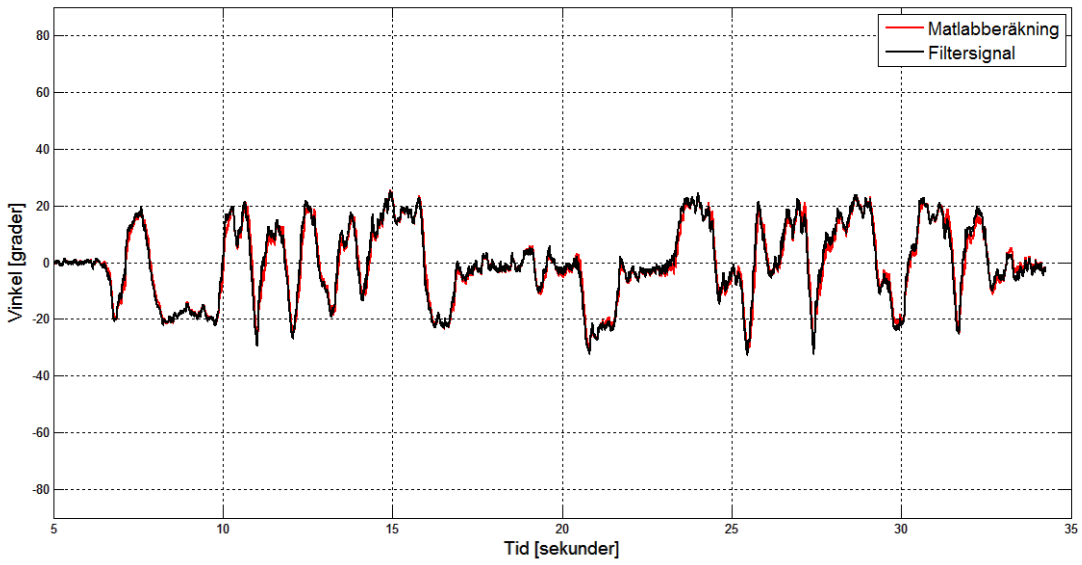
Under utvecklingen användes Processing för att i realtid analysera utsignalerna från sensorerna och filtret under testning i en rigg. När allt verkade fungera samplades värden in i MATLAB för att bekräfta filtrets funktion.



Figur 2.1: Vinkeldata från accelerometer, gyroskop och komplementärfilter vid testning i rigg

I figur 2.1 visualiseras komplementärfiltrets egenskaper tydligt då accelerometerns högfrekventa brus filtreras bort och endast bidrar med dess lågfrekventa förändringar, samt att endast gyroskopets högfrekventa förändringar ger inverkan på utsigalen. Komplementärfiltret fungerade tillfredsställande, vilket gjorde att arbetet med Kalmanfiltret avslutades.

Accelerometersignalen är brusig och för att verifiera att bruset inte uppstått på grund av fel i kommunikationen mellan mikrokontrollern och datorn beräknades nytt komplementärfilterresultat i MATLAB. Datan plottades tillsammans med avläsningen från mikroprocessorn, se figur 2.2. Signalerna sammanfaller vilket visar att mätvärdena stämmer.



Figur 2.2: Verifiering av signaler

2.5 System

Tillsammans med reglersystem och tillhörande hårdvara behövs även andra komponenter, så som batterier, motorer samt motorkontroller och diverse andra kringkomponenter. Val av komponenter och deras funktion beskrivs i följande avsnitt.

2.5.1 Batteri och motor

Tack vare den utveckling som pågått under de senaste åren har Lithium Polymerbatterier (*LiPo*) blivit billigare, kapabla till att leverera högre strömmar samt fått högre kapacitet. De har därför blivit vanliga som strömförsörjning i mindre system som kräver höga strömmar under en relativt lång tid. En nackdel med LiPo-batterier är att om de laddas av en inkompatibel laddare, korsluts eller får fysiska skador, kan det leda till självantändning [11]. Det är därför viktigt att de hanteras varsamt.

Borstlösa DC-motorer (*BLDC-motorer*) är idag en vanlig typ av motor i quadcopterar. Det som skiljer borstade DC-motorer från borstlösa är att lindningarna är statiska, magneterna roterar samt att de har tre faser. På grund av BLDC-motorers uppbyggnad måste en motorkontroller kopplas mellan motorn och strömkällan, som styr ut rätt spänning till de olika faserna. BLDC-motorer har flera fördelar som DC-motorer saknar; vidare varvtalsspann och mindre mekaniskt motstånd tack vare avsaknad av borstar, bättre varvtal/momentkaraktäristik än den för DC-motorer, högre verkningsgrad samt att de är billigare att underhålla och har längre livslängd. Samtidigt har BLDC-motorer sina nackdelar; konstruktionen är mer komplex och därmed dyrare att tillverka samt att de kräver en motorkontroller som blir en tilläggskostnad [12]. En annan nackdel är

att systemet består av två systemkomponenter i serie, vilket gör att risken för att hela systemet fallerar är högre än om det bara finns en systemkomponent som utgör hela systemet.

Elektroniken som används för att styra en BLDC-motor kallas för motorkontroller, eller Electronic Speed Controller (*ESC*). Den innehåller en mikrokontroller som styr en trefasinverterare. Kommunikationen mellan mikrokontroller och motorkontrollern sker via en PWM-signal (pulsbreddsmodulering), där pulsens varaktighet motsvarar ett visst varvtal från motorn. En funktion i mikroprocessorns kod bestämmer pulsens varaktighet. Med delarna som används i detta projektet motsvaras motorernas hela varvtalsspann av ett signalspann mellan 0 och 18000.

2.5.2 Propellrar

För att omvandla den roterande energin som fås ut av motorerna till kraft används propellrar, som håller quadcoptern i luften. Propellrar har oftast två stycken parametrar; diameter och stigning. Parametrarna uttrycks oftast såsom (diameter x stigning) i enheten tum, se tabell 2.1. Diametern är måttet rakt över propellerbladen, och stigningen anger hur lång sträcka en propeller idealt tar sig genom en trög fluid om den roterar ett varv. Propellrar till mindre radiostyrda system tillverkas oftast av plast, trä eller kolfiber.

2.5.3 Komponenter

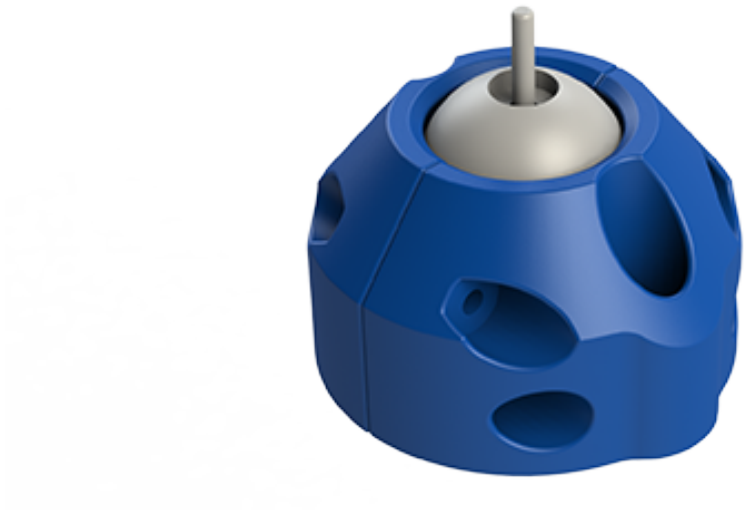
Nämnt i avsnittet syfte (1.4) riktar sig rapporten mot reglersystem, modellering och mjukvara. För att underlätta för de delarna behövs en stabil konstruktion som tål små krascher som kan uppstå under tester. Därför köptes färdiga komponenter som antogs klara de kraven. Utöver kravet ett klara små krascher behövdes komponenter tas fram med kravet att klara lyfta. Uppskattning gjordes att lyftkraften bör vara minst dubbel så stor som quadcopterns tyngd. Att välja komponenter utefter det kravet kan vara problematiskt då totalvikten baseras på valet av komponenter. Uppskattningar och förstudier kombinerades och sammanställdes. Vald motor, se tabell 2.1, hade en sammanställning med lyftkraft beroende på propeller och batterival. På så sätt kunde lyftkraften uppskattas och vara till grund för ytterligare komponentval. Utifrån sammanställningen kan komponentvalet ändras eller kompletteras.

Tabell 2.1: Valda delar

Del	Fabrikat	Modell	Beskriving
Ram	Turnigy	Talon	Tillverkad av kolfiber
Motor	HobbyKing	NX-4006	530kv BLDC
Sensor	Analog Devices	ADXL345	Accelerometer
Sensor	Honeywell	HMC5883L	Magnetometer
Sensor	STMicroelectronics	L3G4200D	Gyroskop
Styrkort	Microchip	PIC32	32-bitars mikrokontroller @ 80MHz
Motorkontroller	Turnigy	Multistar	20 Ampere brushless ESC
Batteri	ZIPPY	Flightmax	2800mAh, 4 celler och 30C
Propeller	HobbyKing	10x4.5	Med- och motursrotation
Sändare	Turnigy	9X	9-kanaler 2.4GHz
Mottagare	Turnigy	9X	9-kanaler 2.4GHz

2.5.4 Egentillverkade komponenter

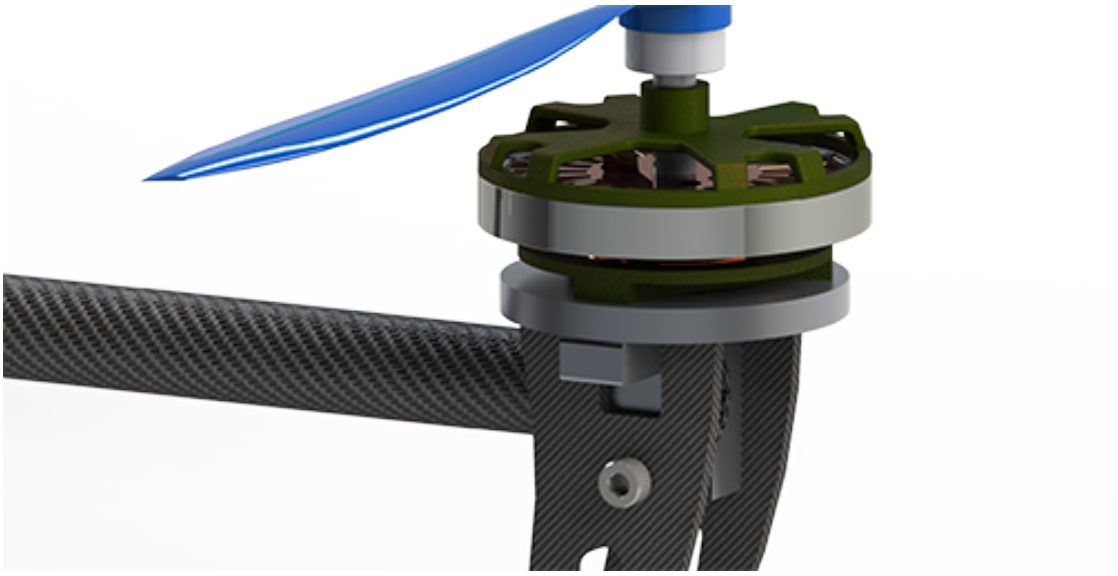
För att på ett enkelt och säkert sätt kunna testa quadcopterns beteende med implementerat reglersystem byggdes en testrigg där antalet frihetsgrader begränsades. Riggen förenklar testerna och minskar även risken för att utrustning eller personer i närheten blir skadade. I grunden bestod den av en stålram med fäste för enkelt byte mellan moduler. Modulerna som tillverkades var i syfte att begränsa quadcopterns rörelse till en frihetsgrad, endast pitch eller roll, eller tre frihetsgrader, pitch, roll och yaw. För test i tre frihetsgrader konstruerades en kulled med hjälp av 3D-printing (se figur 2.3).



Figur 2.3: Kulled

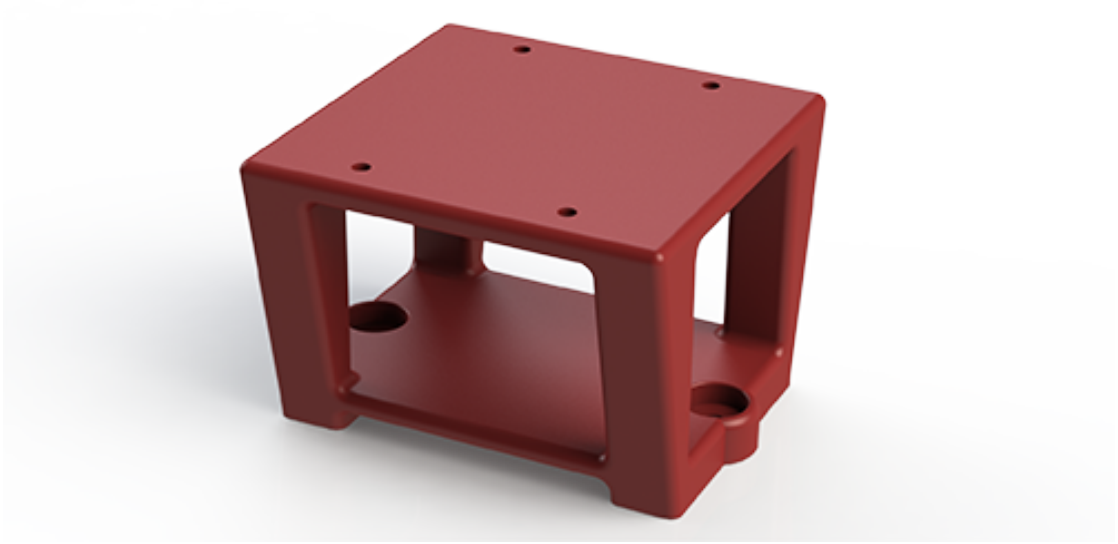
Då de införskaffade komponenterna inte var optimalt anpassade till varandra har det varit nödvändigt att tillverka egna infästningskomponenter för att möjliggöra en optimal montering. Då motorerna ej var möjliga att montera på ramens färdiga motorfästen tillverkades egna motorfästen (se figur 2.4 och 2.5) Dessa tillverkades i aluminium.

Figur 2.4: Egentillverkat motorfäste



Figur 2.5: Egentillverkat motorfäste, monterat

För att möjliggöra friheter i komponentplacering används 3D-utskrift för konstruktion av en komponentkonsoll (se figur 2.6).



Figur 2.6: Komponentkonsol

3

Modellering

För att på ett strukturerat sätt ta fram fungerande reglerparametrar behövs en modell av quadcoptern. Kapitlet beskriver kombinationen av matematisk modellering, tester och 3D-modellering som tillsammans beskriver quadcopterns dynamik. Modellverifikationen bekräftar hur väl ekvationerna överensstämmer med verkligheten.

3.1 Matematisk modellering

Den matematiska modelleringen av quadcoptern kan genomföras utifrån flera olika teorier, men resulterar oftast i samma ekvationer förutsatt en rimlig detaljnivå. Detaljer som kan påverka de slutgiltiga ekvationerna är till exempel försummandet av aerodynamiska effekter. Modelleringen ligger till grund för linjäriseringen som beskrivs i avsnitt 3.3. Utgångspunkten i den matematiska modelleringen är Newtons andra lag samt Eulers rörelseekvation som resulterar i quadcopterns rörelseekvationer.

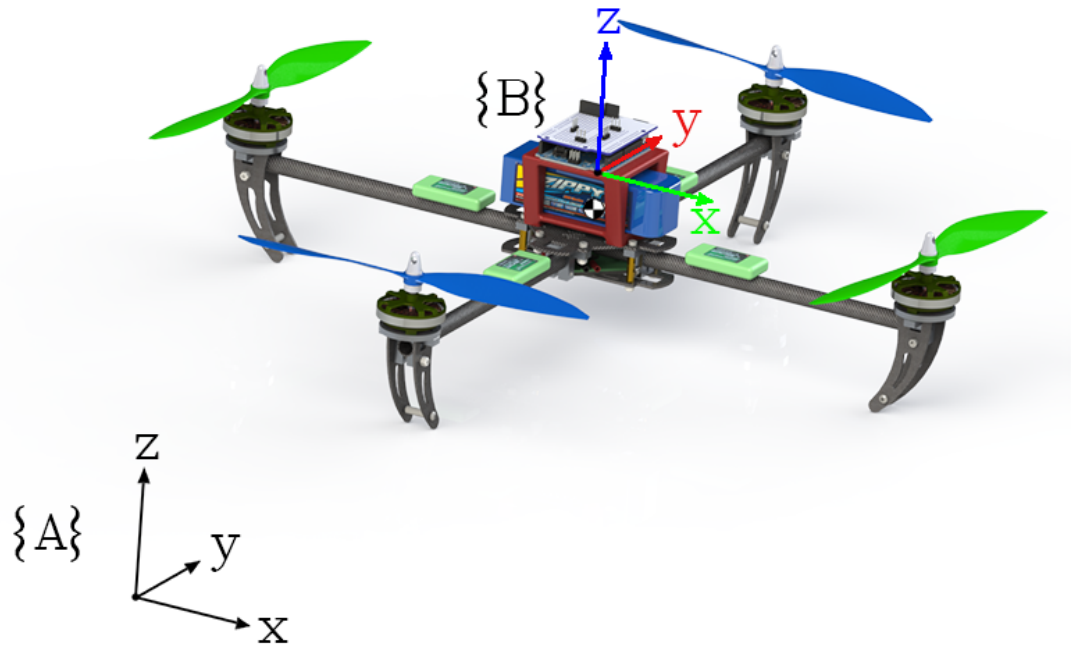
Vid modellering av en quadcopter enligt Newton-Eulers metod definieras först följande ekvationer

$$F = ma, \tag{3.1}$$

$$I\dot{\Omega} = -\Omega \times I\Omega + \tau. \tag{3.2}$$

där F = Kraft, m = Massa, a = Acceleration, I = Tröghetsmatris, Ω = Vinkelhastighetsvektorn och τ = Moment.

Ekvation (3.1) är Newtons andra lag som visar att kraften är lika med massan multiplicerad med accelerationen. Ekvation (3.2) är den generella vektorformen av Eulers ekvationer som anger förhållandet mellan kroppens vinkelacceleration, dess vinkelhastighet och externa moment kring en punkt i det intertiala koordinatsystemet $\{A\}$. Koordinatsystemet är ett fixt koordinatsystem med jorden som referens, se figur 3.1.



Figur 3.1: Illustration av quadcopterns $\{B\}$ och det inertiala koordinatsystemet $\{A\}$

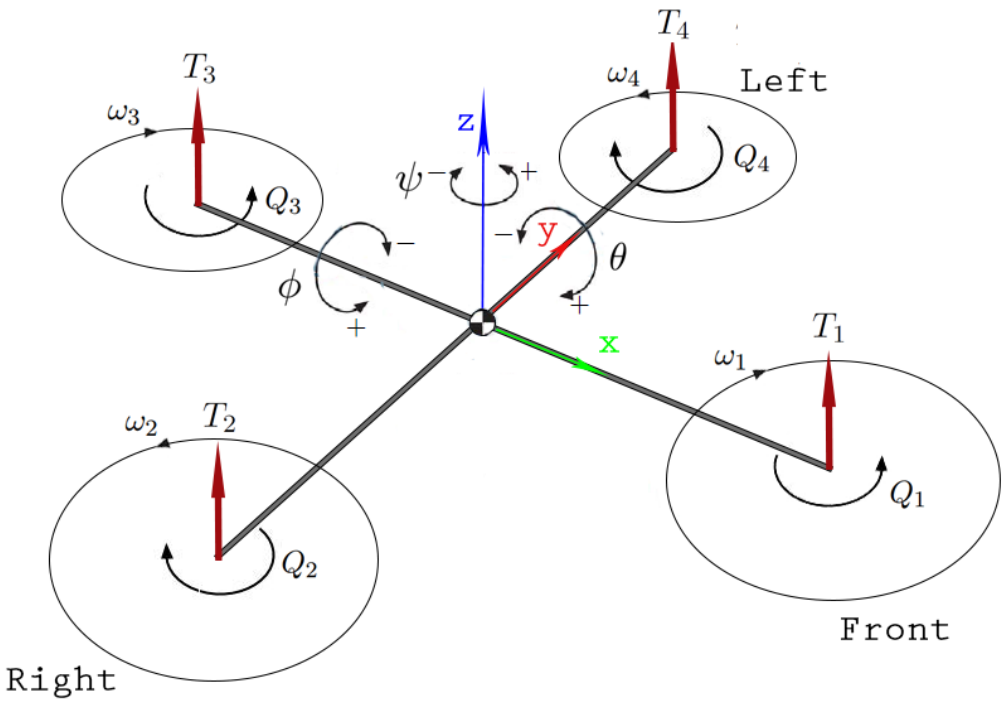
Matrisen I innehåller kroppens tröghetsmoment kring varje axel och definieras enligt

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}. \tag{3.3}$$

Tröghetsmomenten anger det vridmomentmoment som krävs för en given ändring av kroppens rotationshastighet kring en given axel och kan tas fram genom beräkningar för hand eller från en 3D-modell i datorn. För quadcoptern togs tröghetsmatrisen fram med hjälp av programvara för 3D-modellering.

I figur 3.2 nedan definieras de variabler som används i den matematiska modelleringen. Där Q = moment, T = lyftkraft, ω = vinkelhastighet för motor i och (ϕ,θ,ψ) är vinklarna för roll, pitch och yaw.

Första steget i den matematiska modelleringen är att uttrycka quadcopterns läge i rummet, vilket görs med två koordinatsystem. Det första, $\{A\}$, utgår från jorden (inertiala koordinatsystemet) och det andra, $\{B\}$, har quadcopterns centrum som mittpunkt, se figur 3.1. Quadcopterns koordinatsystem anger dess translation och rotation med det inertiala koordinatsystemet som referenspunkt. Genom att relatera koordinatsystemen till varandra kan quadcopterns position uttryckas. För att kunna relatera koordinatsystemen måste först begreppet rotationsmatris utredas.
Den vanligaste metoden för att uttrycka rotationer i tre dimensioner är med Eulervinklar.



Figur 3.2: Quadcopterns dynamik

Metoden går ut på att dela upp rotationerna så att de uttrycks som en rotation kring en enskild axel. Rotationerna uttrycks i 3×3-matriser [13],

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}, \tag{3.4}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \tag{3.5}$$

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.6}$$

där (3.4) är rotation kring x-axeln, (3.5) är kring y-axeln och (3.6) är kring z-axeln. (ϕ,θ,ψ) är vinkeln för roll, pitch och yaw, se figur 3.2. Man kan utifrån dessa matriser

uttrycka vinkelhastighetsvektorn med hjälp av ekvation (3.7), där Ω beskriver vinkelhastigheten kring varje axel.
Vid rotation i tre dimensioner påverkas vinkelhastigheten kring varje axel av både pitch-, roll- och yaw-vinkeln. För att kunna relatera dessa vinklar till det inertiala koordinatsystemet används vektorn Ω. Vektorn är projektionen av roll-, pitch- och yaw-vinklarna på det inertiala koordinatsystemet och definieras enligt [14] som

$$\Omega = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_x(\phi) \left(\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_y(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \right). \tag{3.7}$$

Efter insättning av rotationsmatriserna i ekvation (3.7) erhålles

$$\Omega = \underbrace{\begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix}}_{T} \cdot \dot{\beta}. \tag{3.8}$$

där matrisen T projicerar derivatan av roll, pitch och yaw på det kroppsfixa koordinatsystemet. $\beta = (\phi,\theta,\psi)^T$ motsvarar alltså vinklarna roll, pitch och yaw, se fig 3.2.

Man kan även ta fram en total rotationsmatris, R,

$$R = \begin{bmatrix} c(\phi)c(\theta)c(\psi) - s(\phi)s(\psi) & c(\theta)c(\psi)s(\phi) + c(\phi)s(\psi) & -c(\psi)s(\theta) \\ -s(\psi)c(\phi)c(\theta) - c(\psi)s(\phi) & c(\phi)c(\psi) - c(\theta)s(\phi)s(\psi) & s(\phi)s(\theta) \\ c(\phi)s(\theta) & s(\theta)s(\psi) & c(\theta) \end{bmatrix}. \tag{3.9}$$

Matrisen beskriver hela quadcopterns rotation [13], alltså relaterar rotationsmatrisen det kroppsfixa koordinatsystemet till det inertiala. Matrisen R tas fram genom att multiplicera ihop matris (3.4), (3.5) och (3.6). I matrisen står s för sinus, c för cosinus och vinklarna är för roll, pitch och yaw.

Rotationsmatrisen, R, används för att rotera krafterna från motorerna genom att multiplicera den med kraftvektorn, se ekvation (3.10). Rotorbladen roterar kring z-axeln och därmed har varje motors lyftkraft samma riktning. Rörelseekvationerna för quadcoptern är

$$m\ddot{x} = mg + RT_F, \tag{3.10}$$

$$I\dot{\Omega} = -\Omega \times I\Omega + \tau \tag{3.11}$$

där $\tau = [\tau_1, \tau_2, \tau_3]^T$, $T_F = [0, 0, T_\Sigma]^T$, $x = (x,y,z)^T$ alltså quadcopterns position i det inertiala koordinatsystemet och g är tyngdaccelerationen. τ_1 är momentet kring x-axeln, τ_2 är kring y-axeln och τ_3 är kring z-axeln i $\{\mathrm{B}\}$. T_Σ definieras i ekvation (3.14).

Ekvation (3.11) används senare med (3.8) vid linjäriseringen av systemet, se avsnitt 3.3.

En annan aspekt som måste introduceras för att kunna genomföra den matematiska modelleringen är quadcopterns aerodynamik. Att ta fram en aerodynamisk modellering som stämmer bra överens med verkligheten är dock svårt eftersom man måste ta hänsyn till många okända parametrar. En aerodynamisk effekt som har prioriteras bort är *blade flapping*. För mer information om *blade flapping* refereras läsaren till [15]. För att inte behöva räkna ut de aerodynamiska effekterna kan man ta fram experimentiella konstanter, i detta fall C_T och C_Q, där effekterna ingår. Ett exempel på en aerodynamisk effekt som ingår i konstanterna är den lyfkraft som blåser på ramen och därför motverkar quadcoptern när den lyfter. Det som främst är av intresse med konstanterna är motorernas relation från styrsignal, u, till lyftkraft, C_T, men även relationen mellan styrsignal och vridmoment, C_Q.

$$T_i = C_T u_i \tag{3.12}$$

$$Q_i = (-1)^i C_Q u_i \tag{3.13}$$

T_i = Lyftkraft från den i:te rotorn
C_T = Proportionalitetskonstanten mellan styrsignal och lyftkraft
Q_i = Moment från den i:te rotorn
C_Q = Proportionalitetskonstanten mellan styrsignal och vridmoment
u = Styrsignal
i = {1,2,3,4} representerar en av de fyra motorerna.

Proportionalitetskonstanterna inkluderar även motorernas egenskaper som elektromotorisk spänning (*EMK*) och inre resistans. Då konstanterna baseras på statiska tester tas här ej hänsyn till motorernas dynamiska egenskaper som exempelvis stigtid.

Total lyftkraft vid hovring beräknas som summan av motorernas lyftkraft [16], alltså enligt

$$T_\Sigma = \sum_{i=1}^{4} T_i = C_T \sum_{i=1}^{4} u_i. \tag{3.14}$$

Rotorbladet som är betecknat med en etta i figur 3.3 sitter på ramarmen med längden d, som är fronten på quadcoptern. Vinkeln mellan varje arm och x-axeln i {B} betecknas α_i. För att kunna definiera rotationens riktning på varje motor införs också sgn$(u) \in$ {-1,1}, där +1 är medurs och -1 är moturs. sgn(u) anger tecken på styrsignalen.

Momentet kring de tre axlarna beskrivs som en funktion av kraft från varje motor och beräknas enligt [16] som

$$\tau_1 = C_T \sum_{i=1}^{4} d \sin(\alpha_i) u_i, \tag{3.15}$$

$$\tau_2 = -C_T \sum_{i=1}^{4} d\cos(\alpha_i) u_i, \tag{3.16}$$

$$\tau_3 = -C_Q \sum_{i=1}^{4} sgn(\omega_i) u_i, \tag{3.17}$$

där d är armlängden på ramen. Framtagningen av konstanterna C_T och C_Q beskrivs i avsnitt 3.2.

Efter att ha uttryckt momentet kring varje axel kan rörelseekvationerna skrivas genom att sätta in momenten tillsammans med tröghetsmatrisen och vinkelhstighetsvektorn i ekvation (3.11).

Ekvation (3.14) till (3.17) kan skrivas på matrisform enligt

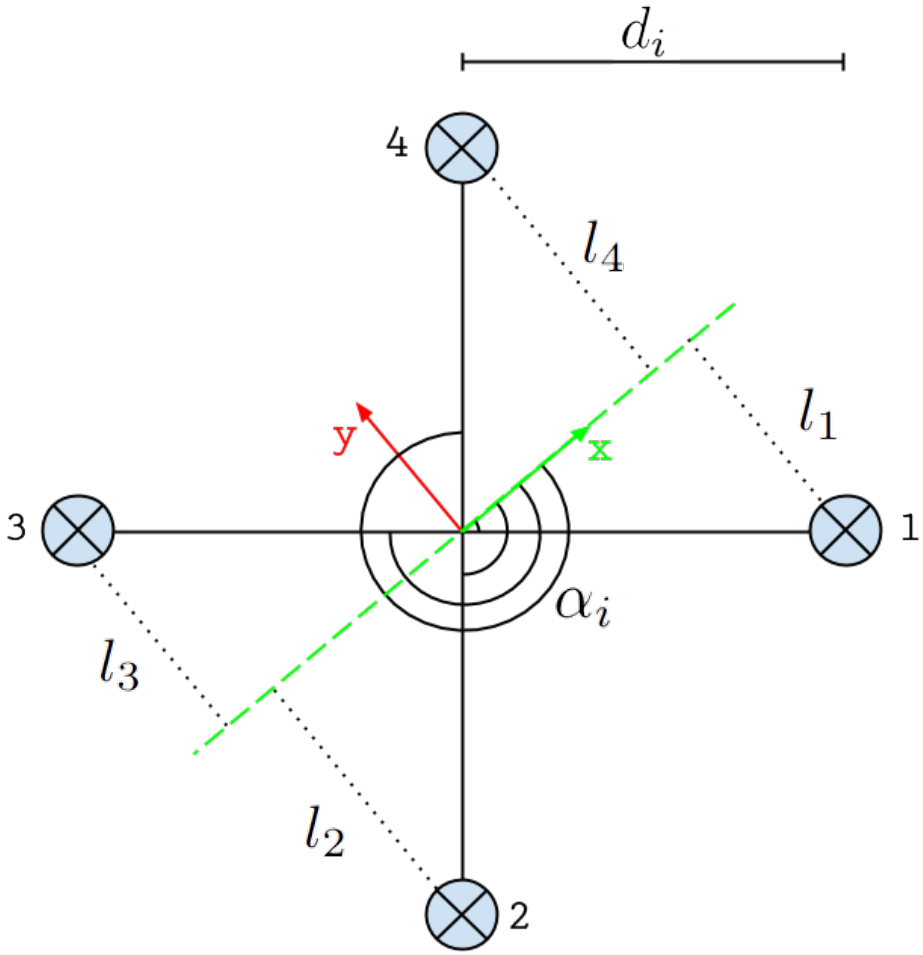
$$\underbrace{\begin{bmatrix} T_\sum \\ \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}}_{E} = \underbrace{\begin{bmatrix} C_T & C_T & C_T & C_T \\ 0 & dC_T & 0 & -dC_T \\ -dC_T & 0 & dC_T & 0 \\ -C_Q & C_Q & -C_Q & C_Q \end{bmatrix}}_{D} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{3.18}$$

För given lyftkraft och givna moment kan erforderlig styrsignal beräknas genom att invertera D-matrisen och multiplicera den med E-matrisen. För att quadcoptern ska hovra beräknas lämplig styrsignal när τ_i=0 och T_Σ= mg.

α i ekvationerna för momenten, som är noll i vårat fall, är vinkeln mellan varje arm på quadcoptern i förhållande till x-axeln i {B}. Vinkeln illustreras i figur 3.3 och definieras enligt

$$\alpha_i = \alpha_0 + i \cdot \frac{\pi}{2}, \qquad i \in [0, 1, 2, 3]. \tag{3.19}$$

$$\tau_1 = C_T du(\sin(\alpha_1) + \sin(180 - \alpha_2) - \sin(\alpha_3 - 180) - \sin(\alpha_4) \tag{3.20}$$



Figur 3.3: Rotationsmoment

Ekvation 3.20 är ekvation 3.15 med alla summationer utvecklade. Vid framtagning av rörelseekvationerna är quadcopterns koordinatsystem låst, alltså är skillnaden mellan x-axeln i {B} och ramarm nummer ett lika med noll. Det betyder att motor ett och tre kommer sammanfalla med x-axeln och därmed inte påverka momentet kring den axeln. Det kan även förklaras genom att betrakta ekvation 3.20 och bild 3.3, där vinklarna α_1=0 samt α_3=180.

Vid beräkning av momentet kring y-axeln kommer motor två och fyra sammanfalla med axeln, vilket medför att de inte bidrar med moment.

3.2 Modellkomplettering

För att linjäriseringen ska stämma överens med systemet, samt för att simuleringarna ska kunna reglera den faktiska kraften eller momentet, snarare än motorsignaler, krävs ett samband mellan dessa. Därför genomfördes ett lyftkraft- och momenttest. Testerna kommer dessutom att fastställa de två återstående parametrarna i den matematiska modellen, C_T och C_Q.

För att få fram förhållandet mellan motorsignal och motorernas moment användes bara en motor för att förenkla testet. Motorn i fråga monterades vinklat så att ett horisontellt luftflöde och ett vertikalt motormoment erhölls. Quadcoptern var endast fri i en frihetsgrad så den kunde bara vippa i den aktiva motorns led. Undertill, på motsatt sida till den aktiva motorn, placerades en våg. Genom att stega motorerna från minimum till maximum och notera utslagen på vågen kunde förhållandet mellan moment och motorsignal beräknas.

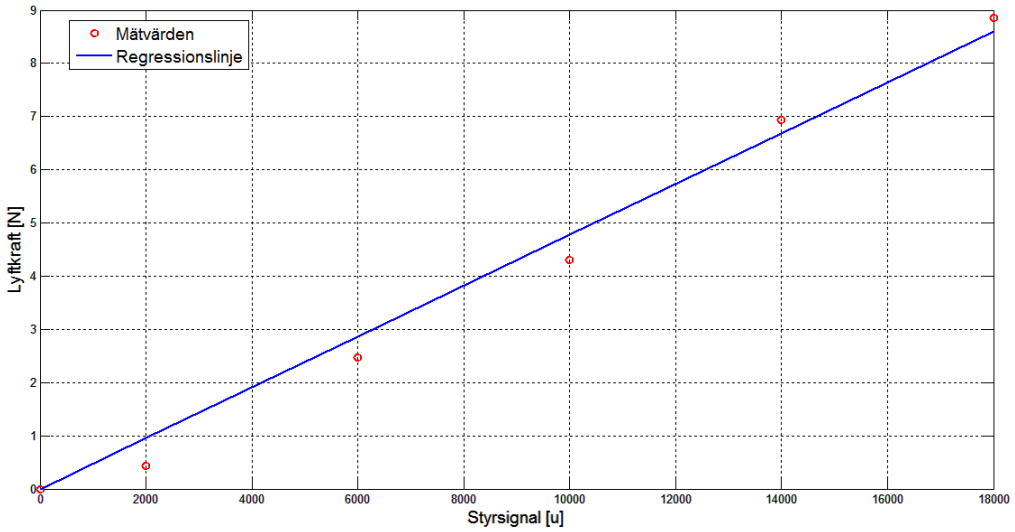
Lyftkrafttestet gick till på liknande sätt med den enda skillnaden att den aktiva motorn stod vertikalt, det vill säga, i sitt ursprungsläge. Motorerna stegades samtidigt som vikten noterades och förhållandet mellan lyftkraft och motorsignal erhölls.

Utifrån mätdata, krafterna F_m och styrsignalerna S_m, kunde en regressionslinje (räta linjens ekvation, som antas passera igenom origo) derivata approximeras med

$$\hat{k} = \frac{\sum S_m \cdot F_m}{\sum S_m^2}. \tag{3.21}$$

Med samma ekvation kan regressionslinjens derivata för momenttestet beräknas.

Resultat Genom att utföra tester kan styrsignal relateras till motsvarande kraft eller moment från motorerna. Anledningen till de två testerna, var att göra den simulerade modellen så realistisk som möjligt. Resultatet för lyftkraftstestet kan ses i figur 3.4.



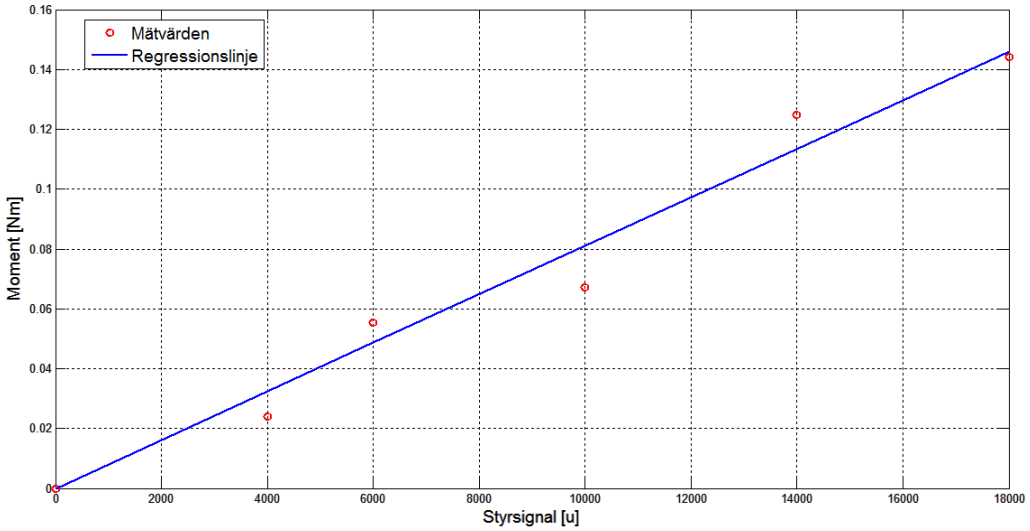
Figur 3.4: Lyftkraft vid olika styrsignaler samt regressionslinje

Utifrån data som testerna gav kunde en funktion tas fram som beskriver förhållandet mellan styrsignalerna för motorna och deras resulterande lyftkraft enligt

$$T = C_T \cdot u = 4.7751 \cdot 10^{-4} \cdot u, \tag{3.22}$$

där u är styrsignalen till respektive motor och T blir kraften som motorerna skapar. Förhållandet antas vara linjärt, och de avvikelser som uppstår i mätdatan antas bero på på grund av onoggrannheter hos mätinstrumentet.

På liknande sätt genomfördes ett test för att få förhållande mellan styrsignal och momentet för att kunna rotera quadcoptern kring z-axeln, alltså styra yaw. Resultatet kan ses i figur 3.5.



Figur 3.5: Moment vid olika styrsignaler

Även här antogs förhållandet vara linjärt sådant att följande funktion relaterar styrsignal till rotationsmoment

$$Q = C_Q \cdot u = 8.1045 \cdot 10^{-6} \cdot u, \tag{3.23}$$

där u är styrsignalen till motorkontrollern och där Q är momentet som motorerna åstadkommer.

3.3 Linjärisering

För att ta fram ett snabbt och robust reglersystem krävs en modell av processen. De metoder som användes för beräkning av reglerparametrar fungerar endast för linjära system. Detta blir problematiskt då en quadcopters dynamik är olinjär. Det kan man dock kringgå genom att linjärisera systemet kring en arbetspunkt, och konstruera regulatorn till att fungera kring just den arbetspunkten. Kraven för att det skall fungera är att modellen för quadcoptern stämmer väl överens med verkligheten samt att den under flygning inte avviker för mycket från den linjäriserade punkten.
Givet ett dynamiskt system

$$\dot{x} = f(x,u) \tag{3.24}$$

så ges en linjär approximation av systemet enligt

$$\Delta\dot{x} = A\Delta x + B\Delta u \tag{3.25}$$

där A och B är Jacobianmatriser av tillståndsekvationer deriverade med avseende på respektive tillståndsvariabler så att

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6}{\partial x_1} & \cdots & \frac{\partial f_6}{\partial x_6} \end{bmatrix}, B = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6}{\partial u_1} & \cdots & \frac{\partial f_6}{\partial u_6} \end{bmatrix}. \tag{3.26}$$

De ekvationer som beskriver quadcopterns alla tillstånd och som därför har använts vid linjäriseringen är tidigare nämnda i avsnitt 3.1 men presenteras på nytt enligt följande:

$$\dot{\Omega} = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = I^{-1}(\tau - (\Omega \times I\Omega)) \tag{3.27}$$

$$\dot{\beta} = \begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = T^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{3.28}$$

Det tillstånd som quadcopten borde fungera i som bäst är då den hovrar, det vill säga då den inte är vinklad. Därför har detta tillstånd används som punkt vid linjärisering enligt:

$$x_0 = \begin{bmatrix} p_0 \\ q_0 \\ r_0 \\ \dot{\theta}_0 \\ \dot{\phi}_0 \\ \dot{\psi}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.29}$$

där p, q och r är systemets vinkelhastigheter kring x, y och z samt $\dot{\theta}$, $\dot{\phi}$ och $\dot{\psi}$ är tidsderivatan av pitch, roll och yaw.
Matrisen A (3.26) räknas fram genom att derivera ekvationerna (3.27) och (3.28) enligt

$$\begin{bmatrix} \dot{f}_1 \\ \dot{f}_2 \\ \dot{f}_3 \end{bmatrix} = \frac{\partial \dot{\Omega}}{\partial x}, \begin{bmatrix} \dot{f}_4 \\ \dot{f}_5 \\ \dot{f}_6 \end{bmatrix} = \frac{\partial \dot{\beta}}{\partial x}. \tag{3.30}$$

På liknande sätt räknades inputmatrisen B för systemet fram. Det som skiljer sig är att

tillståndsekvationerna istället deriveras med avseende på styrsignalsvektorn

$$u_0 = \begin{bmatrix} u_{\theta_0} \\ u_{\phi_0} \\ u_{\psi_0} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.31}$$

som även den är lika med noll vid hovring. Här är u_θ, u_ϕ och u_ψ skillnaden mellan motorernas styrsignaler i respektive vinkel.

Resultat

Jacobianmatrisen A och inputmatrisen B i den givna linjäriseringspunkten:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} B = \begin{bmatrix} 30.120 \cdot 10^{-3} & -4.332 \cdot 10^{-5} & 1.997 \cdot 10^{-6} \\ -4.332 \cdot 10^{-5} & 30.010 \cdot 10^{-3} & -8.594 \cdot 10^{-7} \\ 2.883 \cdot 10^{-5} & -1.241 \cdot 10^{-5} & 1.118 \cdot 10^{-3} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.32}$$

Matrisen för utsignalerna, (3.33), innehåller mätningarna av tillståndsvariablerna från sensorerna och ser ut på följande vis:

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.33}$$

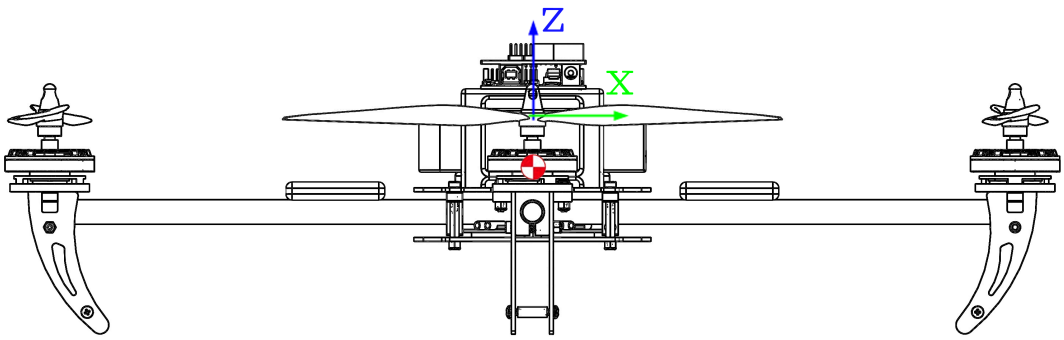
D-matrisen antas vara lika med noll för att det inte existerar någon direkt koppling mellan insignal och utsignal. Genom att använda dessa matriser kunde överföringsfunktionerna för respektive vinkel tas fram. Kommandot $G(s) = C(sI_{id} - A)^{-1}B$ i MATLAB konverterar systemet från tillståndsrum till ett system av överföringsfunktioner. Här står I_{id} för identitetsmatrisen. Nedan presenteras överföringsfunktionerna för respektive vinkel. Funktionerna användes vid simulationen av systemet.

$$G_\theta = \frac{30.120 \cdot 10^{-3}}{s^2}, G_\phi = \frac{30.010 \cdot 10^{-3}}{s^2}, G_\psi = \frac{1.118 \cdot 10^{-3}}{s^2}. \tag{3.34}$$

3.4 3D-modellering

Vid utvecklingen av quadcoptern har 3D-modeller främst använts för att få en visuell bild av quadcoptern och då kunna optimera komponentplacering. Med en visuell bild av ingående delar ges en bredare förståelse för hur komponenter sammanfogas och monteras vilket minimerar risker för framtida problem vid fysisk konstruktion. 3D-modellen har även haft betydelse vid utveckling av nya komponenter. 3D-modelleringsprogramvaror ger tillgång till materialdatabaser vilket ger möjligheten att simulera verkligheten och utföra realistiska virtuella tester. Med modelleringsprogramvaror kan matematiska parametrar som är nödvändiga för den matematiska modelleringen så som tröghetsmoment beräknas. Utöver dessa parametrar har komponentplacering betydelse för quadcopterns stabilitet.

Med en 3D-modell kan komponenter flyttas för att uppnå optimala förhållanden. De mest relevanta parametrarna rörande quadcopterns geometri är tröghetsmoment och tyngdpunkt i förhållande till koordinatsystemet i propellrarnas horisontalplan (se figur 3.6).

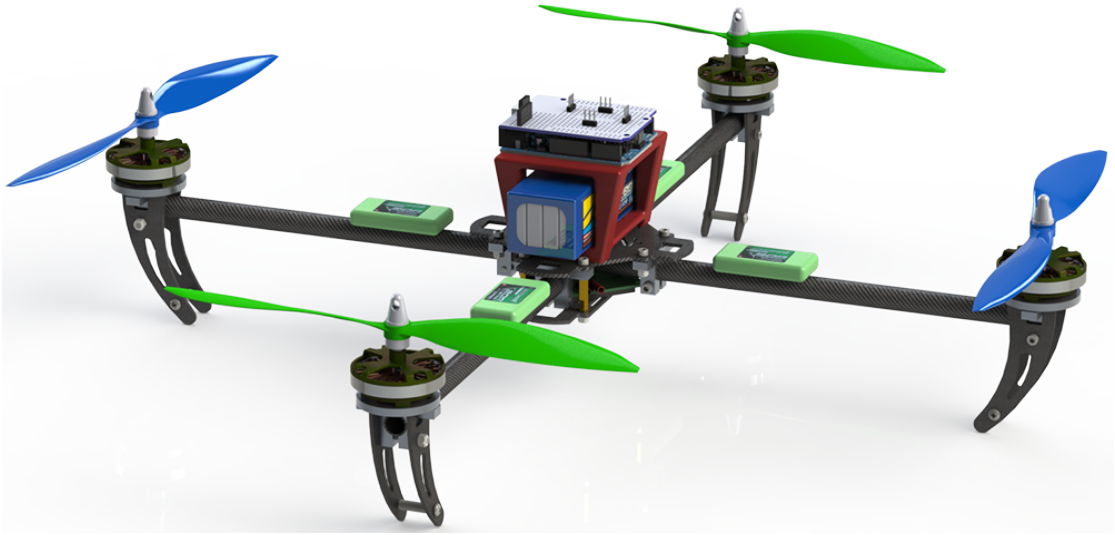


Figur 3.6: Illustration av tyngdpunkt och koordinatsystem

För att uppnå snabb respons, alltså lägre tröghetsmoment i förhållande till koordinatssystemet, figur 3.6, bör tyngdpunkten placeras i koordinatssystemets origo. Ett lägre tröghetsmoment ger dock upphov till att små lyftkraftspådrag på en motor relativt motstående motor ger snabba vinkelförändringar, därmed låg fysisk dämpning. Placeras tyngpunken ovanför porpellrarnas horisontalplan, (positiv z-riktning) bidrar det till ett instabilt system, likt en inverterad pendel. För att inte begränsa möjligheten för framtida kringuttrustning, exempelvis en kamera, bör ingående komponenter placeras på quadcopterns ovansida. Då batteriets vikt är 82% av den totala vikten av alla komponenter förutom fasta komponenter som ram, motorer och propellrar så har batteriets placering stor inverkan på tröghetsmomenten. Därmed bör batteriet placeras nära koordinatsystemet som visas i figur 3.6.

Alla ingående komponenter har modellerats så likt verkligheten som möjligt. Utöver geometriska mått har alla komponenter vägts och modellen har manipulerats för att få komponenternas verkliga massor.

Resultatet av 3D-modelleringen visas i fiugur 3.7.



Figur 3.7: 3D-quad med Turnigy Talon av Alex Baval

Modellen visualiserar inte kabeldragningar eller kabelkontakter. Då dessa är dragna symmetriskt på quadcoptern kan de tas hänsyn till genom att addera dess massor och spridas symmetriskt över ramen.

3D-modelleringen har lett till en modell mycket lik den fysiska quadcoptern och därmed resulterat i en rimlig tröghetsmatris. Med denna konfiguration och med koordinatsystemet i figur 3.6 som referens resulterade det i följande tröghetsmoment:

$$I = 10^{-9} \begin{bmatrix} 4533866.90 & 5647.18 & 6554.37 \\ 5647.18 & 4548817.99 & 5045.43 \\ 6554.37 & 5045.43 & 7251919.19 \end{bmatrix} [kg \cdot m^2]. \tag{3.35}$$

Tröghetsmatrisens inverkan i den matematiska modellen utvärderades vid modellverifikation och simuleringar i SimMechanics. Detta visade att små ändringar i tröghetsmatrisen inte gav märkbara förändringar i simuleringen. Värden som inte tillhör matrisens diagonal var tillräckligt små för att kunna försummas. 3D-modellen är därmed fullt tillräckligt detaljerad för verklighetstrogna resultat.

3.5 Simulering i SimMechanics

För att kunna kontrollera att reglersystemet och quadcopterns dynamik överensstämmer med verkligheten utfördes två olika simuleringar i SimMechanics. Till skillnad från de andra simuleringarna, baseras modellen i Simmechanics inte på en linjär modell, utan numeriska lösningar av de ekvationer som systemet resulterar i.

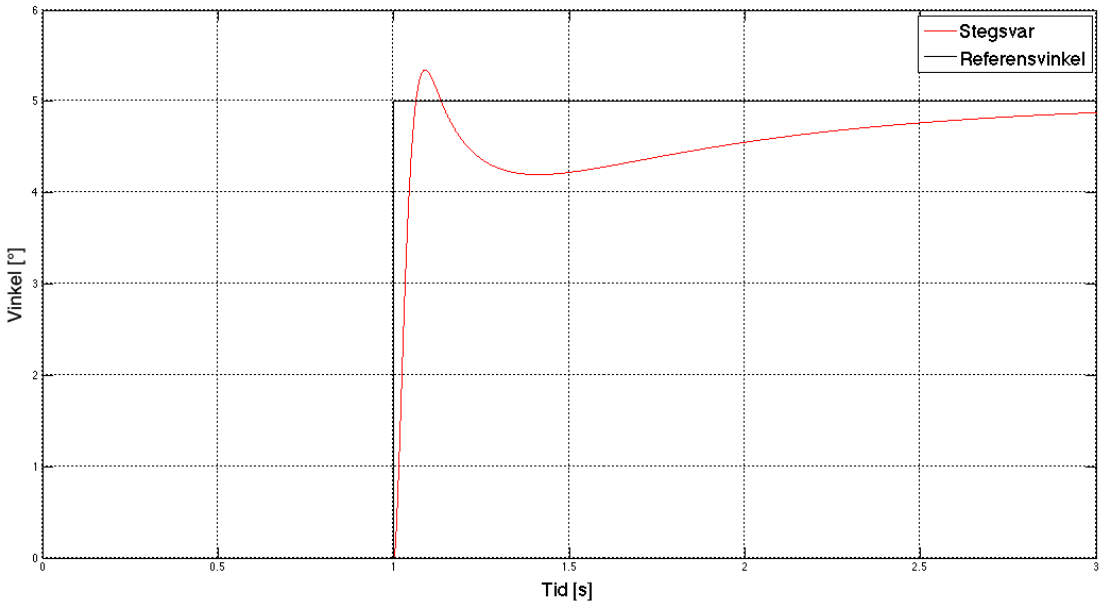
Den första simuleringen representerade testet av en axel i riggen, se bild 3.10, alltså pitch- eller roll-vinkeln. För att kunna utföra simuleringen illustrerades systemet för quadcopterns dynamik med hjälp av block, se figur i appendix C.2. Därefter angavs regulatorparametrar samt referensvinkel. Vinkeln plottades sedan mot tiden för att se hur väl reglersystemet balanserar upp quadcoptern.

Vid den andra simuleringen monterades quadcoptern på kulleden. Detta medförde att quadcoptern kunde röra sig i ytterligare två frihetsgrader, alltså både pitch, roll och yaw. Blockillustrationen av den andra simuleringen kan betraktas i appendix C.3.

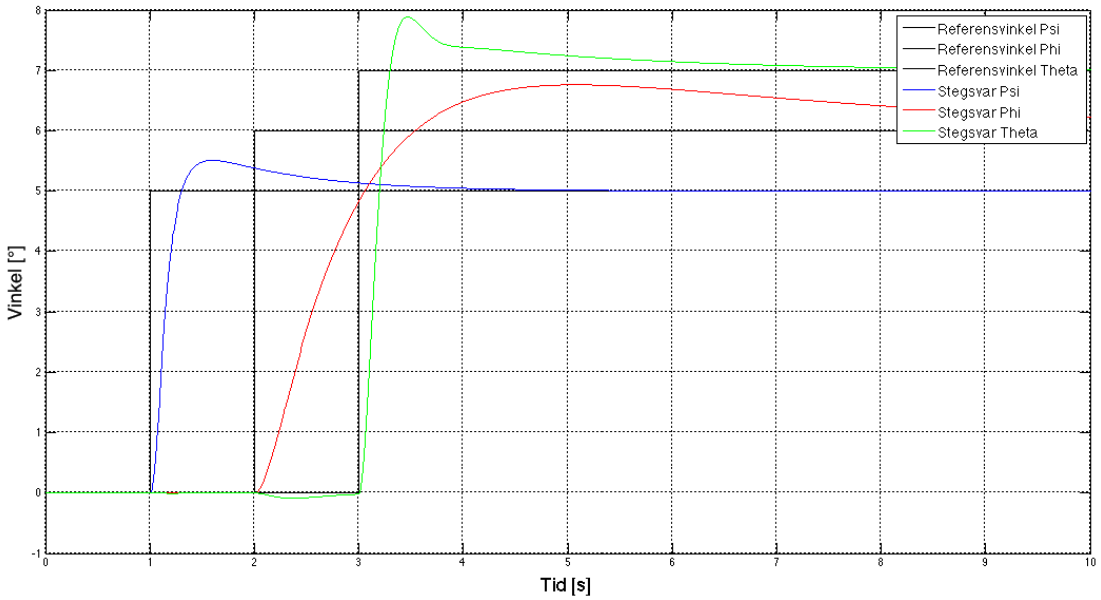
Reglerparametrarna som användes vid simuleringen togs fram med hjälp av en inbyggd funktion i PID-blocket. Funktionen [17] analyserar och pertuberar systemet som PID-blocket är kopplat till, och kalibrerar parametrar utefter en linjär modell som skapas.

Resultat

Simuleringarna som utfördes resulterade i två plottar. Den första då quadcoptern endast hade en frihetsgrad, se figur 3.8, och den andra när den hade tre frihetsgrader, 3.9.



Figur 3.8: Simulering i en frihetsgrad



Figur 3.9: Simulering i tre frihetsgrader

3.6 Modellverifikation

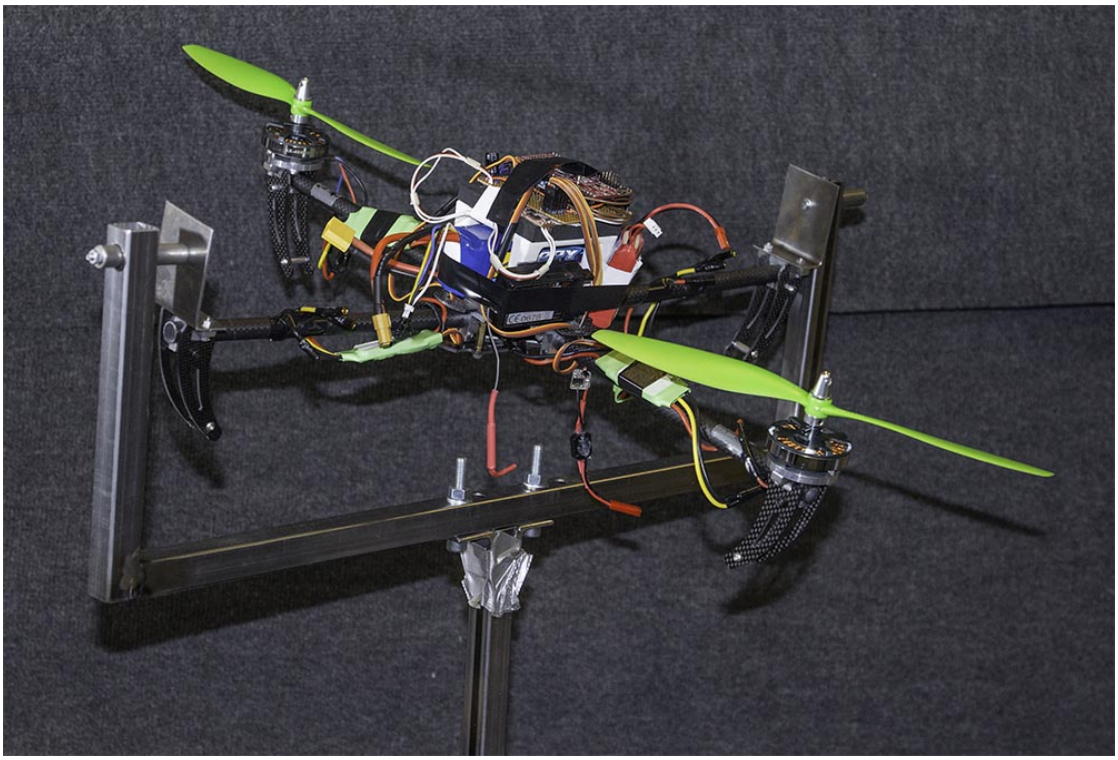
När reglersystem för dynamiska system skall tas fram behövs en korrekt matematisk modell av verkligheten. För att kontrollera att den matematiska modellen motsvarar verkligheten på ett bra sätt, utförs en modellverifikation [18]. Modellverifikationen utförs genom att analysera ett system utan återkoppling och pertubera systemet med olika signaler och observera dess transientsvar.

För att verifiera den matematiska modellen utfördes en simulering i SimMechanics. Simuleringen jämfördes med ett verkligt test vilket användes som referens. Om datan, som plottas för enklare visualisering och jämförelse, stämmer överens i det simulerade testet och verkliga testet visar det att modellparametrarna stämmer.

För att utföra det fysiska testet sattes quadcoptern i en testrigg, se figur 3.10, som begränsade dess rörlighet till en frihetsgrad. Med reglersystemets hjälp balanserades quadcoptern i en fix position fram tills tiden $t = 0$, varpå reglersystemet kopplades ur, styrsignal behålls, och en stegsignal skickades till en av motorerna. Vinkeldatan ut från komplementärfiltret samplades fram tills quadcoptern nått en given slutvinkel. Testet återupprepades fyra gånger för att bekräfta att testets utfall var repeterbart.

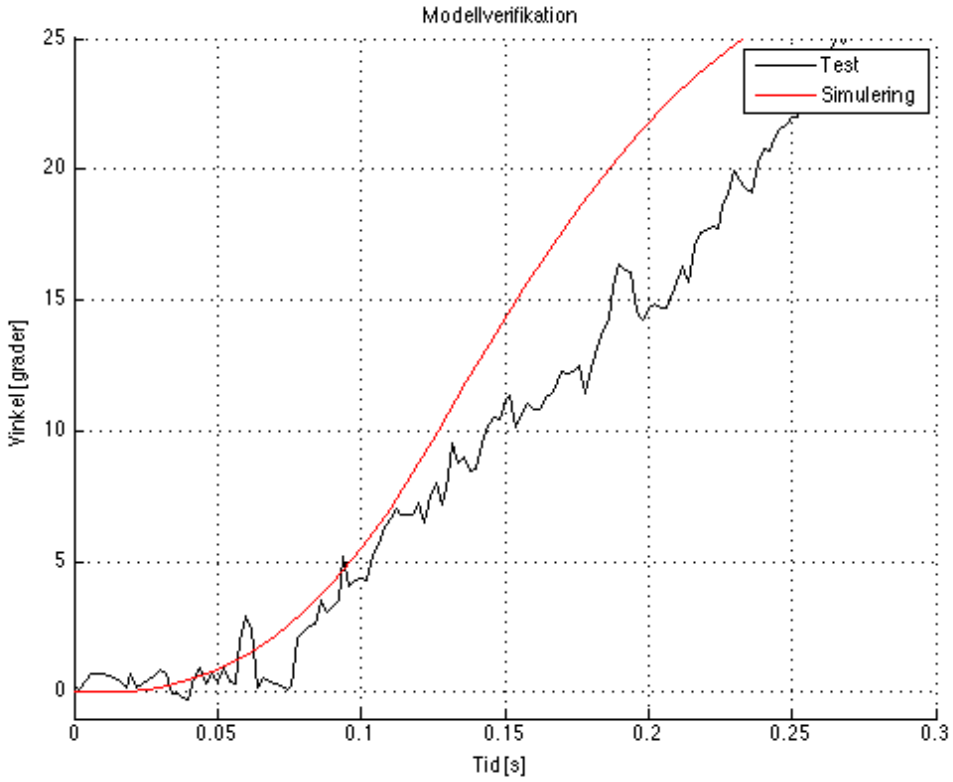
Simuleringen i SimMechanics genomfördes enligt samma metodik som det fysiska testet. Alltså stängdes regulatorn av när quadcoptern balanserade kring jämviktsläget och mer lyftkraft lades på en av motorerna. Innan simuleringen i SimMechanics illustrerades systemet för quadcopterns dynamik med hjälp av block, se appendix C. När hela systemet var uttryckt i block lades även block in för att stänga av regulatorn och öka styrsignalen till en av motorerna vid en viss tidpunkt. Datan från simuleringen analyse-

rades sedan i MATLAB tillsammans med datan från det fysiska testet.



Figur 3.10: Test en axel

Resultat Modellverifikationen resulterade i en plott, se figur 3.11, som jämför det verkliga testet med simuleringen i SimMechanics. Plotten visar hur quadcopterns vinkel varierar under en viss tid och därmed hur väl den matematiska modellen stämmer överens med verkligheten.



Figur 3.11: Modellverifikation

Det fysiska testet som utfördes följer det simulerade fallet väl, enligt figuren, de första 0.1 sekunderna. Då reglersystemet har en sampeltid på 2 μs har 50 beräkningar utförts under den tid då modellen fortfarande stämmer överens med verkligheten.

4

Reglerdesign

Quadcoptern är ett underaktuerat *MIMO-system* (Multiple Input Multiple Output) och behöver därmed ett reglersystem för att kunna flyga stabilt. Det finns flera olika metoder för att reglera dynamiska system; *PID* (Proportional Integral Derivative), *LQR* (Linear Quadratic Regulator), eller *MPC* (Model Predictive Control) är några exempel.

4.1 PID/PDf regulatorn

PID-regulatorn är vanlig inom reglertekniken och använder sig av tre element som viktas med varsin reglerparameter. Den består av en proportionerlig del, en integrerande del samt en deriverande del. PID-regulatorn tar en linjär kombination av dessa elementens verkan av ett reglerfel som används för beräkning av styrsignaler. I de förstudier som gjorts har det visats att vid reglering av en quadcopter används sällan integralverkan [19], och därmed utesluts denna. Alltså kommer utvecklingen bestå av en regulator med en proportionell del och en derivatadel. Dessa delar viktas med deras respektive konstanter K_p och K_d sådan att

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt}, \tag{4.1}$$

där $u(t)$ är beräknad styrsignal och $e(t)$ är skillnaden mellan önskad utsignal och systemets faktiska utsignal. I (4.2) kan ekvationen ses i Laplacedomänen.

$$U(s) = K_p E(s) + K_d E(s)s. \tag{4.2}$$

Konstanterna anpassas så att regulatorn i samband med det reglerade systemet får det beteende som önskas [20].

Eftersom inga system är ideala kommer regulatorns insignaler att innehålla mätbrus som förstärks vid regulatorns derivativa del. För att undvika detta fenomen implementeras ett lågpassfilter i serie med regulatorns deriverande element. Därav förkortningen

PDf-regulator. På så sätt erhålls den nya konstanten T_f för filtret. De tre regulatorerna för vinklarna beräknas med

$$U_\alpha(s) = K_{p_\alpha} \cdot (\frac{T_{d_\alpha} s + 1}{T_{f_\alpha} s + 1}) \cdot E_\alpha(s), \quad \forall \alpha \in \{\theta, \phi, \psi\}, \tag{4.3}$$

där $E_\alpha(s)$ är skillnaden mellan utsignal och önskad referens enligt

$$E_\alpha(s) = R_\alpha(s) - Y_\alpha(s), \quad \forall \alpha \in \{\theta, \phi, \psi\}. \tag{4.4}$$

Utvecklingen av reglersystemet inleddes genom att separera varje enskild axel för att sedan kunna sammanfoga dessa till ett komplett system. Quadcoptern är som nämnts tidigare ett MIMO-system, men approximeras som tre separata *SISO*-system (Single Input Single Output) med hjälp av den linjärisering som utfördes. Exempelvis om en styrsignal ges till motor ett och tre påverkas vinkel θ positivt respektive negativt. Samma princip gäller för vinkel ϕ, men då med motor fyra och två. För att få utslag på vinkel ψ är det istället summan av alla rotorernas moment.
Utifrån denna teorin kan krafterna för respektive motor beskrivas som

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & -1 & -1 \\ 1 & -1 & 0 & 1 \\ 1 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} T_\Sigma \\ U_\theta(s) \\ U_\phi(s) \\ U_\psi(s) \end{bmatrix}, \tag{4.5}$$

där T_Σ står för quadcopterns totala lyftkraft.

Tillämpning

Regulatorparametrarna beräknas genom att utgå från Nyquist-kriteriet. Metoden utför pol- och nodplacering av det linjära systemet. Istället ansätts fasmarginal (φ_m) och överkorsningsfrekvens (ω_c) så att systemet blir stabilt. Det görs i frekvensplanet där $s = j\omega$. För att beräkna förstärkningen K_p nyttjas det att absolutbeloppet av det öppna systemet $\mid L(j\omega_c) \mid = 1$, sådan att

$$\mid L(j\omega_c) \mid = \mid F(j\omega_c) \cdot G(j\omega_c) \mid = 1, \tag{4.6}$$

där $F(j\omega_c)$ och $G(j\omega_c)$ är överföringsfunktionerna för regulatorerna samt vinklarna. För att beräkna tidskonstanten T_d används sambandet mellan systemets fasvinklar enligt

$$\angle F(j\omega_c) + \underbrace{\angle G(j\omega_c)}_{-\pi} - \varphi_m = -\pi, \tag{4.7}$$

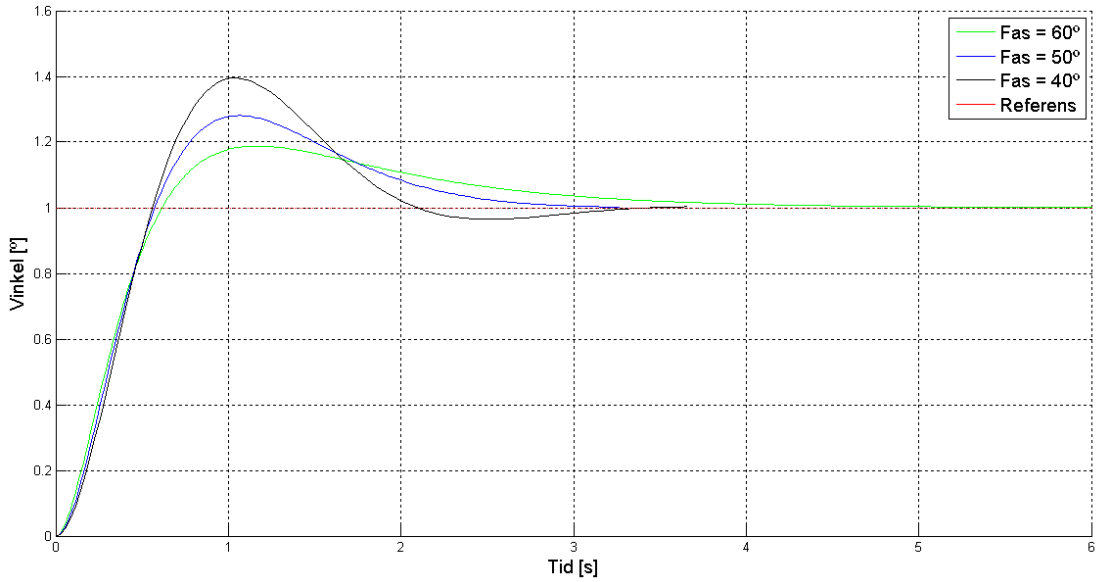
där fasmarginalen, φ_m, enligt teorin bör läggas kring 50° [21]. För att beräkna tidskonstanten T_f nyttjas det att regulatorns fasvinkel i det maximala faslyftet är lika med noll enligt

$$\frac{\delta \angle F(j\omega_c)}{\delta \omega_c} = 0. \tag{4.8}$$

Det som kvarstår i designen av regulatorn är placeringen av systemets brytvinkelfrekvens ω_c. Brytvinkelfrekvensen har följande förhållande med stigtiden för det återkopplade systemet

$$\omega_c \approx \frac{1.386}{t_r}, \tag{4.9}$$

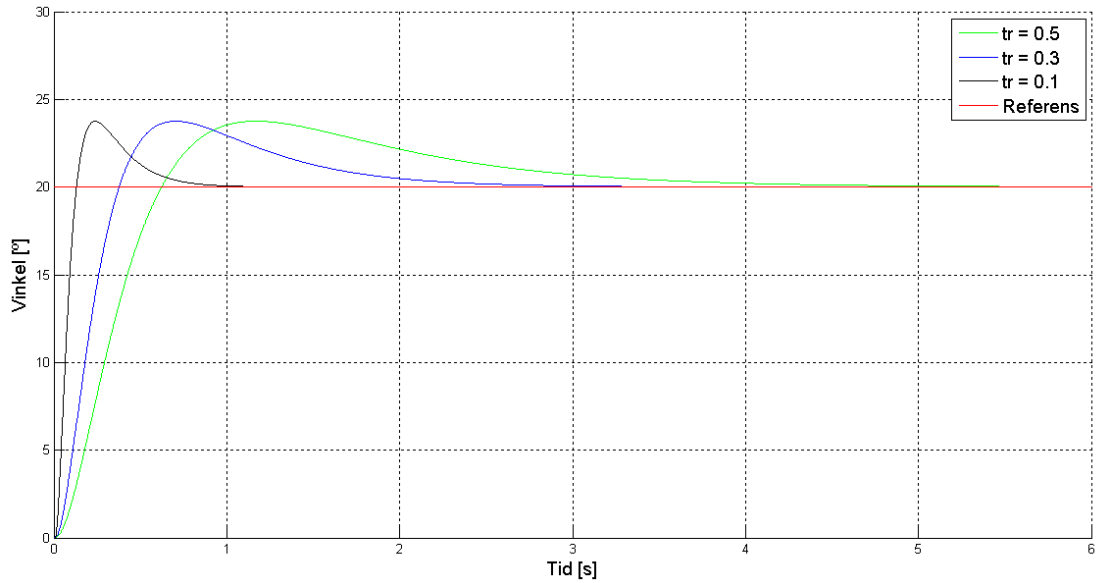
där stigtiden, t_r, väljs utefter hur snabbt systemet bör svara på referenssignaler [22]. För att analytiskt komma fram till vilken kombination av brytvinkelfrekvens (ω_c) och fasmarginal (φ_m) som ger bäst resultat sattes brytvinkelfrekvensen till ett fast värde samtidigt som fasmarginalen lät varieras. Stegsvaret för det återkopplade systemet plottades för varje värde och kan ses i figur 4.1.



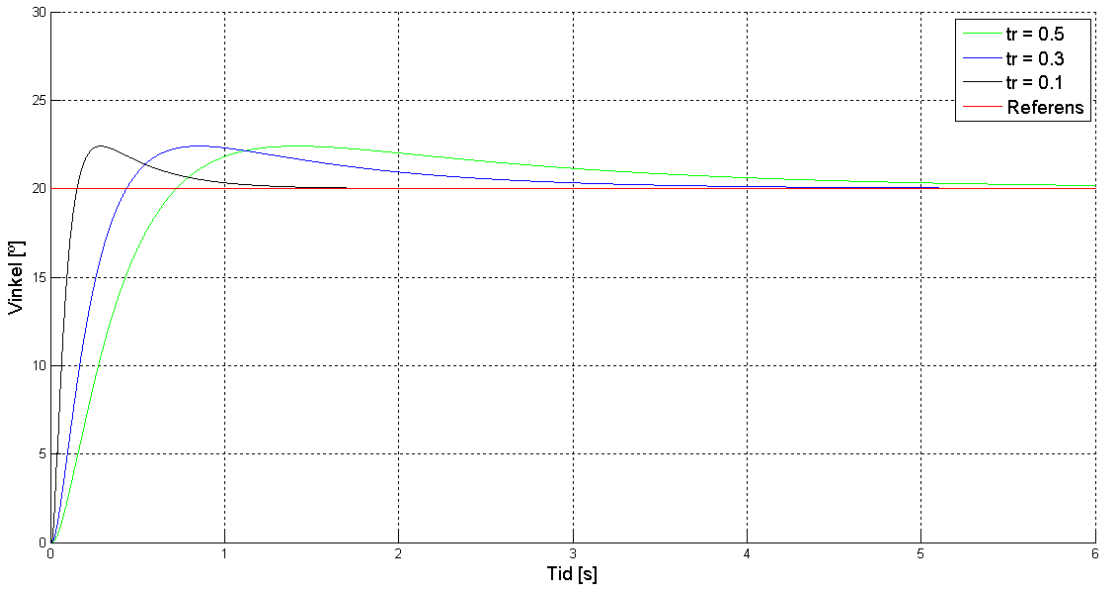
Figur 4.1: Varierande fasmarginal för pitch

Figuren ovan visar simulering av en frihetsgrad. Eftersom pitch och roll har snarlika tröghetsmoment antogs förhållandena vara likvärdiga. Som kan ses i grafen så ger en högre fasmarginal bättre karaktäristik, i form av mindre översläng. Efter att ha gjort på samma sätt för yaw insågs det att det blev orimligt att behandla denna vinkel på samma sätt som för pitch och roll. Yaw anses kräva en mer simpel reglering så därför avvaktas vidare simuleringar av denna vinkel. Detta diskuteras mer i kapitel 5.4.

För att illustrera sambandet mellan stigtid och fasmarginal, har stegsvaren för olika kombinationer plottats som kan betraktas i figurerna 4.2 och 4.3.



Figur 4.2: Varierande stigtid vid $\varphi_m = 60°$



Figur 4.3: Varierande stigtid vid $\varphi_m = 70°$

Figurerna visar att en lägre stigtid ger ett snabbare system, men allt för höga styrsignaler fungerar inte i praktiken. För att få en uppfattning av hur styrsignalsaktiviteten varierar noterades denna under simuleringarna. Tabell 4.1 nedan visar den maximala styrsignalsaktiviteten med två olika fasmarginaler och varierande stigtider.

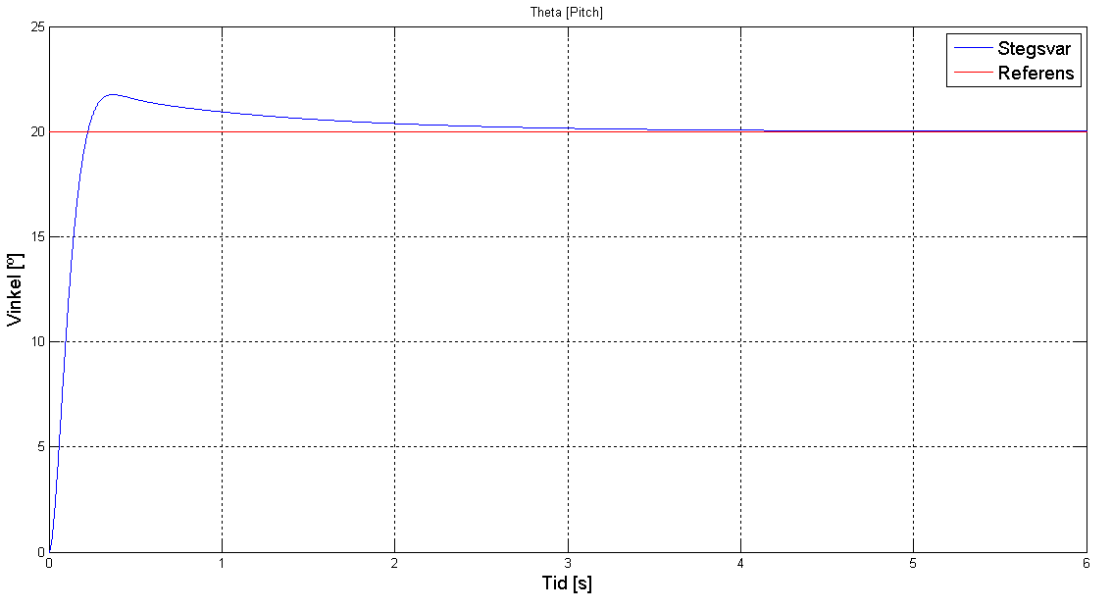
Tabell 4.1: Beloppet av maximala styrsignalen vid varierande t_r

t_r [s]	Styrsignal ($\varphi_m = 60°$)	Styrsignal ($\varphi_m = 70°$)
0.5	< 1500	< 2000
0.4	< 2500	< 3500
0.3	< 4000	< 6000
0.2	< 8500	< 13000
0.1	< 35000	< 50000

Eftersom motorerna har ett styrsignalsspann mellan 0 - 18000, är det inte rimligt att ha alltför höga styrsignaler. Om quadcoptern, i praktiken, flyger med en styrsignal på 9000 till vardera motor och gör en kraftig manövrering är det inte lämpligt med de två nedersta stigtiderna för $\varphi_m = 60°$ eller $70°$ i tabellen ovan. Samtidigt bör stigtiden vara så pass låg så att inga fördröjningar i systemet upplevs av användaren.

Resultat

Fasmarginalen och stigtiden valdes slutligen till $\varphi_m = 70°$ och $t_c = 0.3$, sådan att $w_c = 4.62$, enligt ekvation (4.9). Med dessa värdena erhölls stegsvaret som kan ses i figur 4.4.



Figur 4.4: Stegsvar med $\varphi_m = 70°$ och $t_r = 0.3$ s

Detta motsvarar regulatorparamentrarna som kan ses i figur 4.2.

Tabell 4.2: Kontinuerliga regulatorparametrar för vinkel θ som användes vid simulering

K_p	T_d	T_f
121.2498	1.2276	0.038

Stegsvaret uppfyller de krav som ställts och uppträder tillfredställande. Det vill säga att den har en liten översläng, låg stigtid med en tillräckligt låg styrsignalsaktivitet. Anledningen till att stegsvaret skiljer sig ifrån tidigare plottar är att det är simulerat med en inkluderad baslyftkraft.

4.2 Tidsdiskret reglering

Design av reglersystem sker ofta i den kontinuerliga tidsdomänen. Eftersom inga digitala system kan arbeta i kontinuerlig tid krävs konvertering mellan kontinuerlig tid och diskret tid. Laplacetransformens motsvarighet i diskret tid är Z-transformen. Att gå mellan Laplacedomänen och Z-domänen kallas bilinjär transform [23]. En allmän överföringsfunktion i Z-domänen kan skrivas som

$$H(z) = \frac{n_0 + n_1 \cdot z^{-1} + ...n_n \cdot z^{-n}}{d_0 + d_1 \cdot z^{-1} + ...d_m \cdot z^{-m}}. \tag{4.10}$$

En allmän regel för Z-transformen är att multiplikation med z i Z-domänen motsvarar en tidsförskjutning i den tidsdiskreta domänen

$$\mathcal{Z}^{-1}[H(z) \cdot z] = h[k+1]. \tag{4.11}$$

Motsvarigheten till en tidsförskjutning i Laplacedomänen är multiplikation med e^{sT}

$$\mathcal{L}^{-1}[F(s) \cdot e^{sT}] = f(t+T). \tag{4.12}$$

Därför kan följande likhet definieras

$$e^{sT} = z. \tag{4.13}$$

Löses s ut resulterar ekvationen i

$$s = \frac{1}{T} \cdot \ln(z) \tag{4.14}$$

där T är stegtiden för numerisk integrering i diskret tid. Första ordningens Taylorutveckling av högerledet ger

$$s \approx \frac{1}{T} \cdot \frac{z-1}{z+1}. \tag{4.15}$$

Genom att använda ekvation (4.15) kan s substitueras i den kontinuerliga överföringsfunktionen och ge en diskret överföringsfunktion.

Tillämpning

Den diskreta överföringsfunktionen för regulatorn fås ut genom att använda funktionen c2d() i MATLAB, som tar en överföringsfunktion i Laplacedomänen och systemets samplingstid som inparametrar. Ut från denna funktion fås en överföringsfunktion i z-domänen som inverstransformeras för att få ett uttryck som kan implementeras i ett digitalt system.

Regulatorns överföringsfunktion är förhållandet mellan styrsignal och processfel. Därmed kan regulatorn skrivas på allmän form som

$$F(z) = \frac{U(z)}{E(z)} = \frac{n_0 + n_1 \cdot z^{-1} + ... + n_n \cdot z^{-n}}{d_0 + d_1 \cdot z^{-1} + ... + d_m \cdot z^{-m}}. \tag{4.16}$$

Om respektive nämnare multipliceras upp i motsvarande led fås

$$U(z) \cdot (d_0 + d_1 \cdot z^{-1} + ...d_m \cdot z^{-m}) = E(z) \cdot (n_0 + n_1 \cdot z^{-1} + ...n_n \cdot z^{-n}). \tag{4.17}$$

Om hela uttrycket inverstransformeras fås följande

$$d_0 \cdot u[k] + d_1 \cdot u[k-1] + ... + d_m \cdot u[k-m] = n_0 \cdot e[k] + n_1 \cdot e[k-1] + ... + n_n \cdot e[k-n]. \tag{4.18}$$

Eftersom $u[k]$ är den senaste styrsignalen, löses den ut vilket ger uttrycket för regulatorn

$$u[k] = \frac{-d_1 \cdot u[k-1] - ... - d_m \cdot u[k-m] + n_0 \cdot e[k] + n_1 \cdot e[k-1] + ... + n_n \cdot e[k-n]}{d_0}. \tag{4.19}$$

Om en PDf-regulator med samma struktur som den i ekvation (4.3) konverteras med c2d() i MATLAB fås följande överföringsfunktion

$$\frac{U(z)}{E(z)} = \frac{n_0 + n_1 \cdot z^{-1}}{d_0 + d_1 \cdot z^{-1}}. \tag{4.20}$$

c2d() använder sig av *zero order hold*, vilket innebär att insignalen till systemet, det vill säga felet, antas vara konstant mellan samplingspunkterna. Om den senaste utsignalen för regulatorn löses ut ges

$$u[k] = \frac{-d_1 \cdot u[k-1] + n_0 \cdot e[k] + n_1 \cdot e[k-1]}{d_0}. \tag{4.21}$$

Resultat
När parametrarna för den kontinuerliga regulatorn tagits fram, se tabell 4.3, kunde dessa användas till att ta fram en tidsdiskret regulator.
De parametrar som beräknades från den kontinuerliga regulatorn konverterades till diskret tidskoefficienter med hjälp av c2d() i MATLAB och visas i tabellen nedan.

Tabell 4.3: Kontinuerliga regulatorparametrar för vinkel θ som användes vid stegsvar i rigg

K_p	T_d	T_f
81.89	1.21	0.087

Tabell 4.4: Tidsdiskreta regulatorkoefficienter för Pitch

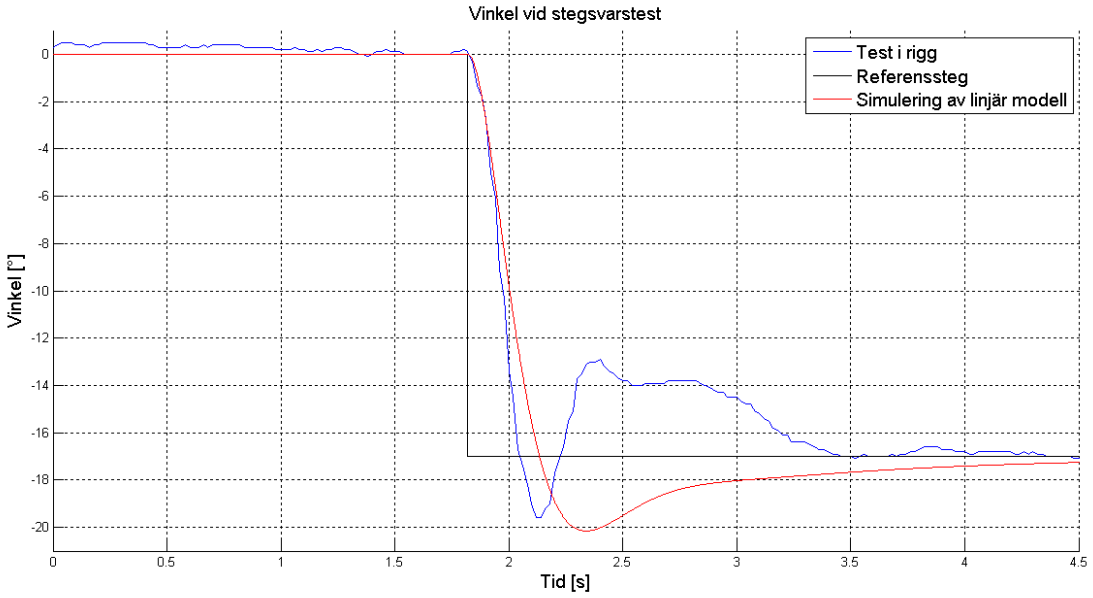
n_0	n_1	d_0	d_1
1139	-1137	1	-0.97727

Quadcoptern testades i riggen med denna regulator. När tester utfördes med de framtagna parametrarna blev systemet instabilt och svängde över. Efter några utförda tester, med olika närliggande parametrar, upptäcktes det att beloppet för parametern d_1 var för stort. Därför justerades denna parameter från -0.97727 till -0.77727 och gav följande tabell:

Tabell 4.5: Justerde tidsdiskreta regulatorkoefficienter för Pitch

n_0	n_1	d_0	d_1
1139	-1137	1	-0.77727

vilket resulterade i stabilitet och en godtagbar stigtid som visas i figur 4.5.



Figur 4.5: Stegsvar för diskret regulator

4.3 Ad hoc-regulator

Vid de flygtest som utfördes först användes inte den framräknade regulatorn vid test. En annan typ av PID-regulator implementerades, benämnd ad hoc-regulatorn i denna rapport. Alla delar räknades ut var för sig och multiplicerades med sina respektive parameter för att sen summeras ihop till den resulterande styrsignalen. Integratordelen är numerisk *zero order hold*-integrering och derivatadelen är Euler-bakåtderivering. Algoritmen såg ut på följande sätt

$$u_\alpha[n] = K_p \cdot e_\alpha[n] + K_i \cdot h \cdot \sum_{i=0}^{n} e_\alpha[i] + K_d \cdot \frac{e_\alpha[n] - e_\alpha[n-1]}{h}, \tag{4.22}$$

där h är periodtiden för uppdateringsfrekvensen av reglersystemet och

$$\forall \alpha \in \{\theta, \phi, \psi\}.$$

Det uppstod dock problem när referenssignalen användes i derivatadelen; avläsningen av referenssignalen var inte helt perfekt och innehöll sporadiska spikar. Detta ledde till att derivatadelen blev väldigt hög och resulterade i ett okontrollerbart beteende. Därför användes så kallat *derivative of measurement* [24], som endast innehåller derivatadelen av utsignalen. Felet i ett återkopplat system definieras som $e = r - y$, vilket gör att tecknet på derivatan är negativt. Eftersom derivatan av utsignalen (vinkeln) mäts direkt av gyroskopet, användes gyroskopavläsningen istället för diskret derivering av beräknad vinkel från filtret, vilket resulterade i följande ekvation:

$$u_\alpha[n] = K_p \cdot e_\alpha[n] + K_i \cdot I_\alpha[n] - K_d \cdot \dot{\alpha}[n]. \tag{4.23}$$

För att få fram någorlunda bra parametrar innan flygning, testades denna regulator i riggen som byggts, för att se vilka parametrar som gav snabb stigtid men samtidigt bra stabilitet.

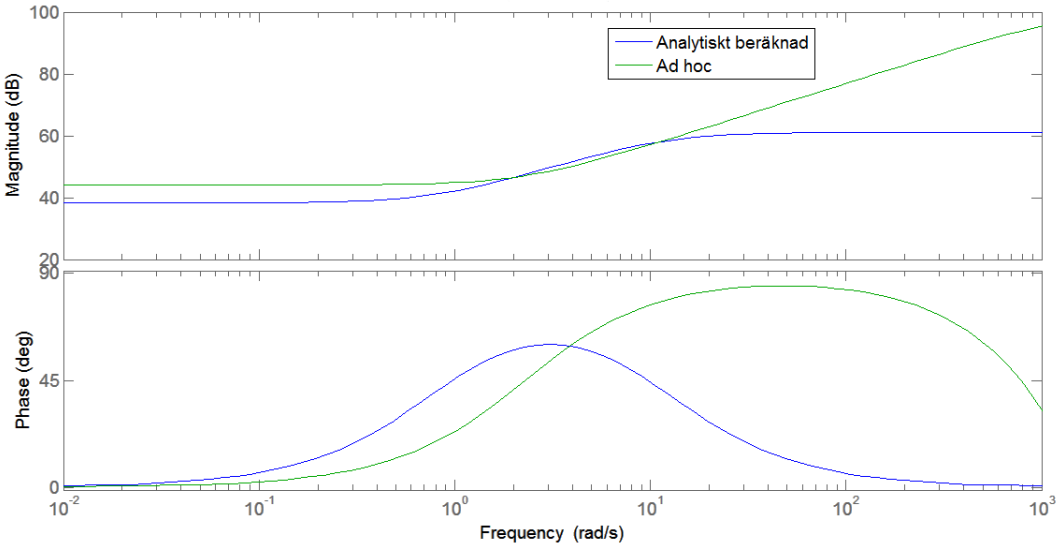
Resultat

Ad hoc-regulatorn var den regulator som användes vid flygning, då den gav bra resultat i riggen. Parametrarna som användes visas i följande tabell:

Tabell 4.6: Tidsdiskreta regulatorkoefficienter för Pitch

Vinkel	P	I	D
Pitch	160	0	70
Roll	160	0	70
Yawrate	120	0	0

Eftersom yaw inte är kritisk för stabiliteten, användes endast en P-regulator för den. Integratordelen implementerades ej då det inte fanns tillräckligt med tid för tester av den. Med denna regulator kunde quadcoptern flygas av en något erfaren pilot. Den var inte lika stabil som kommersiella quadcoptrar. Med fortsatt kalibrering är det möjligt att uppnå högre stabilitet och manövrerbarhet.
Ett bodediagram togs fram för att jämföra Ad hoc-regulatorn och den beräknade kontinuerliga regulatorn i frekvensdomänen



Figur 4.6: Bodediagram för Ad hoc-regulatorn och kontinuerlig regulator

Bodediagrammet visar på att det finns en viss likhet mellan de olika regulatorerna. Bortsett från skillnad i amplitud vid olika frekvenser, finns det även en karaktäristisk skillnad; den analytiskt beräknade regulatorn övergår till en konstant förstärkning vid högre frekvenser, medan Ad hoc-regulatorns förstärkning ökar konstant vid högre frekvenser.
Slutligen utfördes tester för att undersöka det framtagna systemets prestanda. Den kunde hålla sig i luften utan problem och gick att manövrera. Det finns fortfarande vissa svagheter i reglersystemet som gör att den är svårare att flyga än andra kommersiella quadcoptrar.

5

Diskussion

I projektets inledning fanns det tvivel om att alla delar skulle hinna falla på plats innan arbetets slut. Det var många olika ämnen och delar att sätta sig in i som inte bara skulle fungera var för sig, utan även tillsammans. Flera olika personers programkod, lödningar, egentillverkade komponenter, inköpta komponenter och så vidare skulle kombineras till en produkt, vilket i slutfasen visade sig fungera.

Att följa rapporten från början till slut beskriver delarna som krävs för att gå från ingenting, till en flygande quadcopter. I quadcopterns nuvarande stadie finns det dock en del områden som kan utvecklas mer.

5.1 Modellverifikation

När man betraktar figur 3.11, som illustrerar skillnaden i vinkel mellan den verkliga och den simulerade quadcoptern, ser man att skillnaden ökar med tiden. Den matematiska modellen stämmer väl, som tidigare konstaterats, de första 0.1 sekunderna. Därefter ändras vinkeln i det simulerade fallet snabbare än i det fysiska testet. En faktor som kan leda till denna skillnad är den dynamiska friktionen i riggen. Vi ansåg att friktionen var svår att mäta eftersom den mänskliga faktorn hade spelat en stor roll i experimentet vi hade planerat. En annan anledning till att det fysiska testet avviker kan vara mätstörningar i form av brus, från motorerna, som filtret inte lyckas eliminera.

5.2 Testriggar

En viktig del i projektet var användandet av testriggar. Riggarna underlättade en hel del då testerna kunde utföras på ett kontrollerat och säkert sätt. Däremot insåg vi i ett tidigt skede att utformningen av riggarna inte var tillräckligt genomtänkta. Vi tog inte hänsyn till att rotationspunkten för quadcoptern i riggen borde överensstämma med verkligheten, alltså en rotationspunkt i propellrarnas horisontalplan. De två första

riggarna fungerade på liknande sätt, att quadcoptern balanserade likt en inverterad pendel. Vi fick det ändå att fungera även i dessa fall, men det var svårare än vad det borde och återspeglade inte verkligheten på rätt sätt. Därför byggdes en tredje och sista rigg för att ge möjlighet att optimera reglerparametrar för ett fall som är mer likt en fritt flygande quadcopter.

När testet av en axel fungerade tillräckligt bra var nästa steg att testa reglersystemet i kulleden. Problemet med detta var att rotationspunkterna i de olika riggarna inte låg på samma ställe i förhållande till quadcopterns rotationspunkt när den flyger. Detta medförde att reglerparametrarna som var framtagna, och fungerande i en axel, inte fungerade i kulleden. Vi tog då beslutet att gå direkt på flygtest eftersom regleringen för både pitch och roll fungerade väl efter första testet.

5.3 Antaganden

Att mäta, testa och undersöka allt tar tid. Tidigt insåg vi att allt inte går att göra perfekt, så redan från början fick kompromisser göras i hopp om att slutresultatet ej skulle påverkas märkbart. Delar i den matematiska modelleringen försummades, exempelvis aerodynamik och gyroskopiska effekter samt att friktion i testriggarna försummades. Det är chansningar som grundar sig i tron om att de förlusterna är små nog att de inte ger betydande skillnad för slutresultatet. Det är svårt att mäta och konkret säga att rätt saker försummades. Men då modellverifikationen visade bra resultat anser vi att våra antaganden har fungerat och sparat oss tid som vi har kunnat utnyttja till att slutföra arbetet.

5.4 Reglersystem

I slutändan fick vi endast ad hoc-reglersystemet att flyga. Vi hade möjligtvis kunnat få det beräknade reglersystemet att fungera, om vi haft mer tid för test. Att få någonting att fungera tack vare en vedertagen vetenskaplig metod är alltid bättre än någonting som är framtaget med empiriska tester, och inte har någon bakomliggande vetenskaplig metodik. Vi är dock nöjda med resultatet från det beräknade reglersystemet, eftersom det visade på rätt karaktäristik och bra resultat när det testades i riggen.

Vid framtagning av parametrar till reglersystemet har yaw inkluderats, något som kan anses onödigt då yaw inte är lika kritiskt för quadcopterns stablitet. Sensorerna kan inte heller ge någon exakt avläsning på yaw, eftersom accelerometern inte påverkas av yaw, och gyroskopet mäter vinkelhastighet, vilket leder till ett integrerat fel efter en viss tid. Det är dessutom inte logiskt för användaren att kontrollera yaw. Det vore mer passande att kontrollera yaw rate, alltså vinkelhastigheten för yaw. Eftersom de ekvationer som togs fram och linjäriserades innehöll just yaw, var det svårt att beräkna en bra regulator för yaw rate.

6

Utvärdering

Vid framställningen av projektets grundpelare, den matematiska modelleringen, valdes Newton-Eulers metod. Ett alternativ till Newton-Euler är Euler-Lagrange. Då Euler-Lagrangemetoden främst lämpar sig för dynamiska system med ingående rörliga delar har antaganden gjorts att båda metoderna resulterar i samma ekvationer. Detta efter rekommendationer från kunniga inom området. Detta är inget som under projektets gång verifierats då Euler-Lagrange metod är avancerad och ligger utanför projektmedlemmarnas kunskapsområde. För vidare studier inom området kan det finnas ett värde i att framställa den matematiska modelleringen efter Euler-Lagrange teori för att kunna verifiera de antaganden som tagits. En vidareutveckling av den matematiska modelleringen är också att använda sig av kvaternioner istället för Euler-vinklar.

Fysiska parametrar från den matematiska modelleringen som tröghetsmoment har beräknats med 3D-modelleringsprogramvaror med en detaljrik modell. Det har visat sig att små variationer i tröghetsmatrisen inte haft någon större effekt på vare sig modellverifikation eller reglering. Därmed finns inget intresse i en mer detaljerad 3D-modell.

Den matematiska modelleringen avgränsades med att fysiskt ta fram funktioner för lyftkraft och moment per rotor som tar hänsyn till aerodynamiska, elektriska och mekaniska effekter som är svåra att beräkna analytiskt. Då utrustning och verktyg var en begränsande faktor finns det, vid mer omfattande tester, potential till att få mer exakta resultat.

Simuleringen i SimMechanics var en viktig del i projektet, inte minst vid modellverifikationen. En del vi inte lyckades med var att ta fram reglerparametrar från SimMechanics som även fungerade i verkligheten. Anledningen till att vi inte fick fram fungerande parametrar från SimMechanics var framförallt på grund av tidsbrist. Ett potentiellt utvecklingsområde är därför att göra en mer gedigen simulering, där störningar som vindpustar, gyroskopiska effekter från motorer och vibrationer vägs in.

Som tidigare nämnt så finns det många olika sätt att reglera en quadcopter. Med de tidsramarna som vi har haft var det svårt att komma fram till vilket sätt som hade

gett bäst resultat. Vi bestämde oss därför tidigt att vi skulle utveckla en PD-regulator. Det faktum att vi valde ett reglersystem som gruppen kände sig bekväma med gjorde eventuellt att vi gick miste om smidigare och bättre tillvägagångssätt.

Projektet behandlar två typer av filter, komplementärfilter och Kalmanfilter. Komplementärfiltret valdes att implementeras då Kalmanfiltret ansågs för komplicerat och beräkningstungt. Då tester inte genomförts med Kalmanfiltret är det svårt att uttala sig om hur bra komplementärfiltret är. Kapitel 2.4 visar att komplementärfiltret filtrerar bort oönskat bidrag från använda sensorer och anses därför godtagbart. Ett potentiellt utvecklingsområde är att implementera ett Kalmanfilter istället för ett komplementärfilter.

Utöver accelerometer och gyroskop har även magnetometer diskuterats. Magnetometern skulle i mån av tid implementeras, men då arbetet inte nått längre än tester i rigg och endast några enstaka provflygningar, har inte implementation kunnat motiveras. Quadcoptern är fullt fungerande med endast yawreglering med gyroskopets vinkeldata som återkoppling.

Projektets mål, att quadcoptern ska vara så pass stabil att en person utan tidigare erfarenhet inom RC-flyg ska kunna flyga den med enkla instruktioner, har uppfyllts. Reglersystemet dock är inte tillräckligt bra anpassat efter hur människan uppfattar quadcopterns manövrerbarhet. För att uppnå detta behöver parametrar optimeras efter vad som känns som en kontrollerad flygning vilket kräver ett ytterligare antal tidskrävande flygtester.

Utöver projektmålet har projektet resulterat i en infrastruktur för hur man går från ingenting till en flygande quadcopter. Strukturen är anpassad för att enkelt kunna göra ändringar i den fysiska produkten, omplacera komponenter eller implementera kringutrustning. I 3D-modellen kan exempelvis en kamera läggas till och programvaran ger då nya parametrar som implementeras i simuleringen och resulterar i reglerparametrar anpassade efter den nya produkten.

Litteraturförteckning

[1] P. Lambermont, Helicopters and Autogyros of the World (1958).
URL http://www.aviastar.org/helicopters_eng/bothezat.php

[2] Y. Gao, What makes the quadcopter so great for small drones?
URL http://www.forbes.com/sites/quora/2013/12/23/what-makes-the-quadcopter-design-so-great-for-small-drones

[3] B. Belew, Amazon's unveils new Prime Air quadcopter delivery service, too good to be true? (2013).
URL http://www.examiner.com/article/amazon-s-unveils-new-prime-air-quadcopter-delivery-service-too-good-to-be-true

[4] M. Sharma, Indian Startup Uses Drones To Drop Aid In Flood-Ravaged Areas (2013).
URL http://techcrunch.com/2013/08/20/indian-startup-uses-drones-to-drop-aid-in-flood-ravaged-areas

[5] Processing, A short introduction to the Processing software and projects from the community.
URL http://www.processing.org/overview/

[6] Nationalencyklopedin, accelerometer (april 2014).
URL http://www.ne.se/accelerometer

[7] M. N. Armenise, MEMS Gyroscopes, Springer Berlin Heidelberg, 2011, s. 83–102.

[8] S. Colton, The Balance Filter.
URL http://web.mit.edu/scolton/www/filter.pdf

[9] B. G. Welch, G, An introduction to the kalman filter.
URL clubs.ens-cachan.fr/krobot/old/data/positionnement/kalman.pdf

[10] Pieter-Jan, Reading a imu without kalman: The complementary filter (2013).
URL http://www.pieter-jan.com/node/11

[11] Pros and cons of lithium polymer batteries and nickel metalhydride batteries.
URL http://www.ehow.com/info_8127726_pros-cons-nimh-batteries-lipo.html

[12] Brushless dc motors.
URL http://www.dynetic.com/brushless%20vs%20brushed.htm

[13] A. Boström, Rigid body dynamics, Studentlitteratur, 2013.

[14] A. F. Sørensen, Autonomous Control of a Miniature Quadrotor Following Fast Trajectories, 2010, s. 12.

[15] J. G. Leishman, Principles of Helicopter Aerodynamics, Cambridge University Press, 2006, s. 183.

[16] R. Mahony, Multirotor aerial vehicles - modeling, estimation and control of quadrotor (Augusti 2012).

[17] What plant does the pid tuner see?
URL http://www.mathworks.se/help/slcontrol/ug/what-plant-does-the-pid-tuner-see.html

[18] B. Thacker, et. al., concepts of model verification and validation.
URL http://www.ltas-vis.ulg.ac.be/cmsms/uploads/File/LosAlamos_VerificationValidation.pdf

[19] A. Özrür Kivrak, Design of control systems for quadrotor (December 2006).
URL http://acikarsiv.atilim.edu.tr/browse/156/168.pdf&embedded=true

[20] B. Lennartsson, Reglerteknikens grunder, Studentlitteratur, 2011, s. 326–340.

[21] B. Lennartsson, Reglerteknikens Grunder Formelsamling, Studentlitteratur.
URL https://www.studentlitteratur.se/files/products/7307/formelsamling7307.pdf

[22] C. N. Herrick, Basic Electronics Math, Newnes, 1997, s. 140.

[23] S. Engelberg, Digital Signal Processing, An Experimental Approach.

[24] D. Cooper, Pid control and derivative of measruement.
URL http://www.controlguru.com/wp/p76.html

A

Komma igång

För att komma igång och utföra tester med quadcoptersystemet krävs förkunskaper i C-programmering.

A.1 Steg 1

För att använda samma plattform som använts i detta projekt behövs ett prototypkort med en PIC32 på. Med denna behövs även en programmerare, fördelaktligen PICKIT 3. För att programmera kortet behövs utvecklingsmiljön MPLAB IDE för att författa C-koden och ladda över den till mikrokontrollern. Det behövs även en lämplig *kompilator* till koden. Vi använde oss av MPLAB v8.92 och XC32 1.30. Dessa kan hittas här:
MPLAB: `ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_IDE_8_92.zip`

XC32: `ww1.microchip.com/downloads/en/DeviceDoc/xc32-v1.30-windows-installer.exe`

Skulle någon annan version önskas använda finns alla andra versioner på följande sida:
`http://www.microchip.com/pagehandler/en-us/devtools/dev-tools-parts.html`

Notera att kompatibel release av XC32 måste väljas om annan release skulle väljas.

A.2 Steg 2

Öppna bifogat MPLAB-projekt. Fil med .mcp som filtyp.

A.3 Steg 3

De funktioner som man behöver förstå sig på finns bifogade i appendix B.

A.4 Steg 4

Kompilera projektet.

A.5 Steg 5

Programmera kortet.

B

Mjukvara

B.1 Funktioner

Funktion: initINTERRUPT()
Returnerad: Ingenting
Parametrar: Inga
Utför: Initierar interrupt.

Funktion: initUART()
Returnerar: Ingenting
Parametrar: Inga
Utför: Initierar UART.

Funktion: initPWM()
Returnerar: Ingenting
Parametrar: Inga
Utför: Initerar PWM som används för kommunikation till motorkontrollerna.

Funktion: initPORTS()
Returnerar: Ingenting
Parametrar: Inga
Utför: Initierar portar.

Funktion: initI2C()
Returnerar: Ingenting
Parametrar: Inga
Utför: Initerar I^2C.

Funktion: initACC(int scale)
Returnerar: Ingenting
Parametrar: Max/min värde för sensorn, ±2g, ±4g, ±8g, ±16g
Utför: Initerar accelerometer.

Funktion: initGYRO(int scale)
Returnerar: Ingenting
Parametrar: Max/minvärde för sensorn, ±250dps, ±500dps, ±2000 dps
Utför: Initierar gyroskopet.

Funktion: delayUs(int delay)
Returnerar: Ingenting
Parametrar: Fördröjningen i mikrosekunder
Utför: Fördröjer programmet i det antal mikrosekunder som anges som parameter. Kan endast fördröja tiotals mikrosekunder på grund av att klockvariabeln 'time' har den upplösningen.

Funktion: SetDCOCXPWM(int motorsignal)
Returnerar: Inget
Parametrar: PWM-signalen som önskas styra ut
Utför: Sätter PWM-signal till motorkontrollerna för att styra varvtalet. Notera att 'X' i funktionsnamnet byts ut mot siffrorna 1-4 för att styra respektive PWM-port. Max- och minvärde för vilka signaler som ger respons från motorkontrollerna finns definierat som makrona 'MOTOR_MIN' samt 'MOTOR_MAX'.

B.2 Variabler

Namn: time
Typ: unsigned long int
Syfte: Innehåller antalet mikrosekunder som passerat sen processorn slogs på eller senast återställdes. Upplösning 10 mikrosekunder.

Namn: accRawData[3]
Typ: BYTE
Syfte: Skickas in i getAccData() och lagrar rå sensordata från accelerometer.

Namn: gyroRawData[3]
Typ: BYTE
Syfte: Skickas in i getGyroData() och lagrar rå sensordata från gyroskop.

Namn: accAxis[3]
Typ: int

Syfte: Datan som kommer från accelerometern är 16 bitar per axel. Varje axel lagras först i två variabler, alltså två 8-bitarsvariabler för varje axel. Dessa två variabler sätts sen samman i accAxis[]. Element 0 innehåller x-axeln, element 1 innehåller y-axeln och element 2 innehåller z-axeln.

Namn: gyroAxis[3]
Typ: int
Syfte: Datan som kommer från gyroskopet är 16 bitar per axel. Varje axel lagras först i två variabler, alltså två 8-bitarsvariabler för varje axel. Dessa två variabler sätts sen samman i gyroAxis[]. Element 0 innehåller pitchrate, element 1 innehåller rollrate och element 2 innehåller yawrate.

Namn: accFloats[3]
Typ: float
Syfte: Lagrar nedskalad och kalibrerad data från accelerometern.

Namn: gyroFloats[3]
Typ: float
Syfte: Lagrar nedskalad och kalibrerad data från gyroskop.

Namn: dt
Typ: long int
Syfte: Innehåller den fasta tiden, i mikrosekunder, som säger hur långt ett loopvarv skall vara.

Namn: switched
Typ: int
Syfte: Indikerar om switchen på sändaren är på- eller avslagen. Kan användas som bland annat säkerhetsfunktion om något skulle gå fel.

Namn: x1
Typ: int
Syfte: Lagrar antalet mikrosekunder som mättes upp på rollkanalen.

Namn: x2
Typ: int
Syfte: Lagrar antalet mikrosekunder som mättes upp på thrustkanalen.

Namn: x3
Typ: int
Syfte: Lagrar antalet mikrosekunder som mättes upp på pitchkanalen.

Namn: x4

Typ: int
Syfte: Lagrar antalet mikrosekunder som mättes upp på yawkanalen.

Namn: x5
Typ: int
Syfte: Lagrar antalet mikrosekunder som mättes upp på switchkanalen.

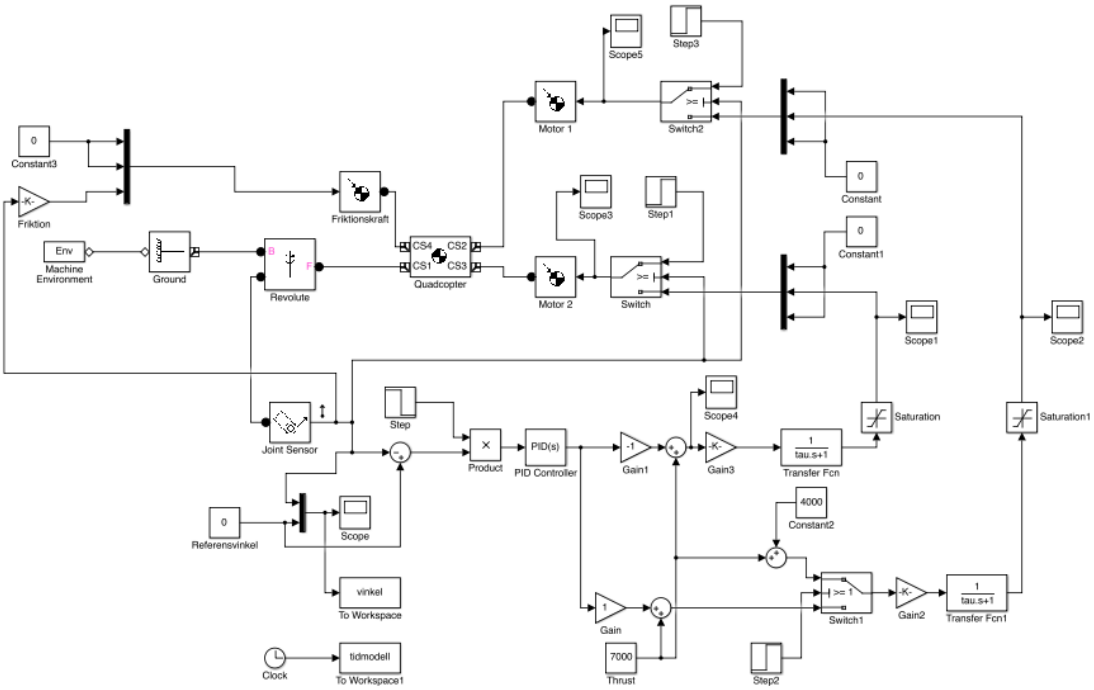
Namn: var1 till var9
Typ: float
Syfte: Parametrar som exempelvis kan användas till tuning av regulatorn. Dessa parametrar kan justeras i realtid under körning av programmet via ett Processingprogram.

Namn: MOTOR_MIN
Typ: Makro
Syfte: Lagrar minsta tillåtna motorsignal för ESC:erna.

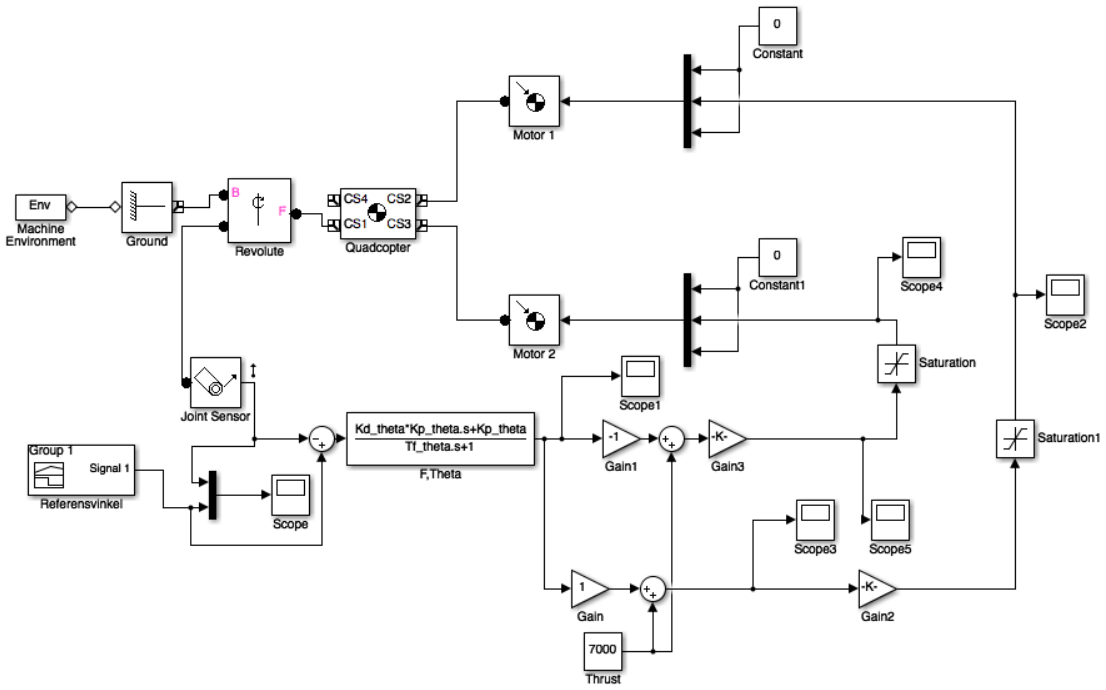
Namn: MOTOR_MAX
Typ: Makro
Syfte: Lagrar högsta tillåtna motorsignal för ESC:erna.

C

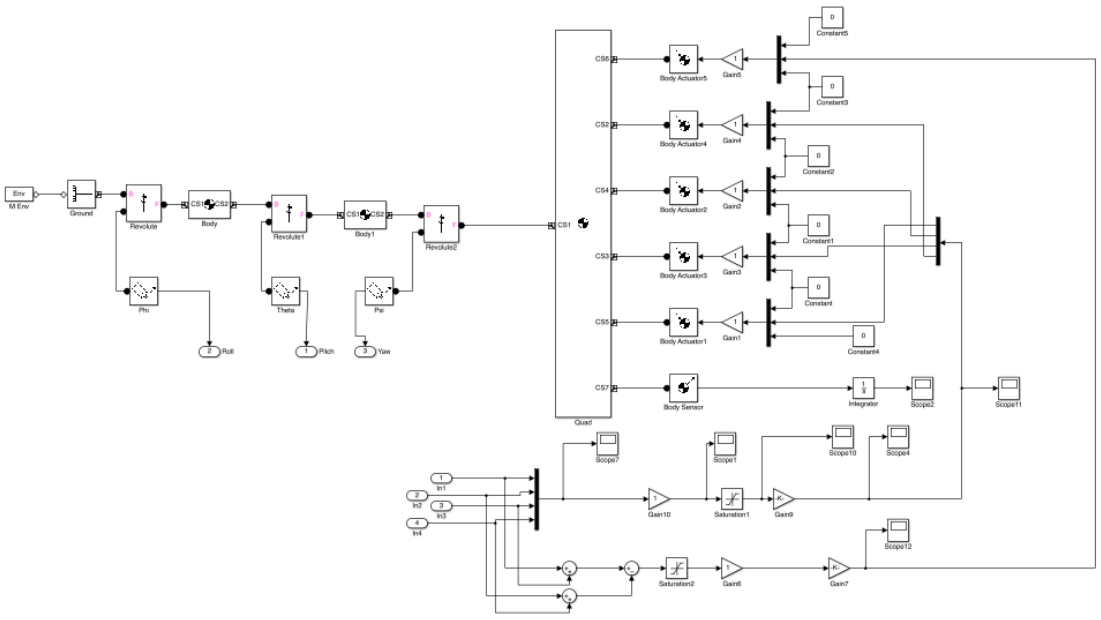
SimMechanics



Figur C.1: Modellverifikation



Figur C.2: Simulering en axel



Figur C.3: Simulering i kulled