Autopilot



Leonardo Sjöberg

EC Utbildning

Examensarbete

202505

# Abstract

This paper is about making an autopilot will the help of a Reinforcement Learning model to make the aircraft maintain level flight. The criterion for this objective is to make the agent control the aircraft within a certain threshold of degrees to make sure that it maintains it level flight in both bank, pitch and heading level. The RL model autopilot gave good result for the objective given, that is that it stayed within a threshold of -+ 5 degrees in bank and pitch, and where even better when looking at the performance of the agent. It also maintained low deviation in heading and made outstanding performance in its objective to make the aircraft level. This paper will also touch on key ideas of how aerodynamics work, as well as how reinforcement learning work. It will also touch on the method on how these results were made, with all the reward functions being explained to why and how they were implemented.

Innehållsförteckning

[Abstract 2](#_Toc197561789)

[1 Inledning 1](#_Toc197561790)

[2 Teori 2](#_Toc197561791)

[2.1 Flygdynamik 2](#_Toc197561792)

[2.1.1 Aerodynamiska grundkrafter 2](#_Toc197561793)

[2.1.2 Manövrerbarhet, kontrollytor och styrprinciper 3](#_Toc197561794)

[2.2 Reinforcement Learning (RL) 3](#_Toc197561795)

[2.2.1 Markov Decision Processes 3](#_Toc197561796)

[2.2.2 Value och policy baserade algoritmer 3](#_Toc197561797)

[2.2.3 Reward funktion och straff 4](#_Toc197561798)

[2.2.4 Mätetal för prestanda 5](#_Toc197561799)

[3 Metod 6](#_Toc197561800)

[3.1 Miljöintegration med MSFS 2024 6](#_Toc197561801)

[3.2 RL-ramverk och modellimplementation 6](#_Toc197561802)

[3.2.1 Definiering av observation, action-space och rewards 6](#_Toc197561803)

[3.2.2 Reward 7](#_Toc197561804)

[4 Resultat och Diskussion 9](#_Toc197561805)

[5 Slutsatser och Diskussion 12](#_Toc197561806)

[5.1 Slutsats 12](#_Toc197561807)

[5.2 Diskussion 12](#_Toc197561808)

[Appendix A 13](#_Toc197561809)

[Källförteckning 15](#_Toc197561810)

# Inledning

Det ökade behovet av autonoma flygsystem och artificiell intelligens inom luftfartsindustrin har under det senaste decenniet drivit fram nya metoder för att förbättra säkerhet, effektivitet och driftsekonomi. Traditionella autopilotsystem baseras ofta på modeller av flygplanets dynamik och fördefinierade regelverk för styrning. Dessa system är välutvecklade och beprövade, men ställs inför utmaningar när de möter oförutsedda störningar eller komplexa flygsituationer som kräver adaptiva lösningar. Samtidigt har framsteg inom maskininlärning, och i synnerhet Reinforcement Learning (RL), visat stor potential att hantera just sådana adaptiva styrproblem genom att låta en agent lära sig optimala strategier via interaktioner med en simulerad miljö.

Inom forskning kring AI-baserade styrning har Reinforcement Learning framgångsrikt applicerats på robotik, spel och fordonsautomation. Inom flygtekniken har RL-modeller föreslagits för uppgifter som landning, vindkompensation och formationsflygning, men få studier har fokuserat på en av de allra grundläggande styruppgifterna, vilket är att hålla ett flygplan i stabil, rak kurs. Att återskapa en autopilot för rak kurs är ett välavgränsat men pedagogiskt rikt problem. Det kräver modellering av flygplanet, utformning av lämplig reward-strukturer och val av algoritmer, samtidigt som det är tillräckligt enkelt för att tydligt illustrera RL-agentens inlärningsmekanismer.

Genom att fokusera på rak kursstyrning kan denna studie bidra med värdefulla insikter i hur Reinforcement Learning-modeller kan klara av klassiska autopilotuppgifter, samtidigt som det möjliggör jämförelser med traditionella kontrollmetoder som PID-regulationer och LQR (Linear Quadratic Regulator). Dessutom blir det möjligt att identifiera vilka aspekter av problemet, till exempel val av observationer (höjd, hastighet, vinkel), actionspace (höjdroderutslag, skevroderutslag) och reward-funktioner (stabilitet, energiförbrukning), som mest avgörande för robust inlärning och kontroll. På så vis kan arbete lägga en grund för vidare studier av mer avancerade styrscenarier, såsom kurvtagning, störningshantering och fullskalig autopilotintegration.

Syftet med denna rapport är att undersöka och återskapa en autopilot för rak kursstyrning av ett flygplan genom användning av en Reinforcement Learning-modell.

1. Kan en Reinforcement Learning-modell styra ett flygplan i rak kurs med en deviation inom 5 grader i roll och pitch?

# Teori

Detta kapitel kommer att ta upp nödvändig teori om Flygdynamik och Reinforcement Learning för att förstå vilka problem som kan förekomma under rak kursflygning, samt hur RL fungerar och hur man ska träna en sådan modell för att uppnå målen som vi sätter på den agenten.

## Flygdynamik

### Aerodynamiska grundkrafter

I rak flygning verkar fyra fundamentala krafter på ett flygplan: lyft, tyngd, drivkraft (thrust) och motstånd (drag). Enligt Studyflight.com (u. å.) så är Lyft den komponent av den aerodynamiska kraften som verkar vinkelrätt mot den relativa luftströmmen och motverkar tyngdkraften. Lyftkraften uppstår genom olika fenomen. En orsak till att lyftkraft genereras är på grund av tryckskillnaderna över vingytornas över och undersida, vilket kan förklaras med hjälp av Bernoullis princip. En annan orsak till lyftkraften beror på Newtons tredjelag som lyder ’varje kraft alltid möts av en lika stor men motsatt riktad kraft’. Denna motsatta kraft induceras av luften som avböjs nedåt från vingen vilket ger en uppåtriktad reaktionskraft som lyfter flygplanet. Lyftkraftens storlek beräknas enligt ekvationen

Där 𝝆 är lufttätheten, V farten relativt till luften, S är vingytan och C\_L den dimensionslösa lyftkoefficienten.

Tyngdkraften är gravitationskraften som alltid verkar nedåt genom flygplanets tyngdpunkt och är produkten av planets massa och tyngdaccelerationen.

Drag är den aerodynamiska kraften som verkar motsatt flygplanets rörelseriktning. Drag uppstår på olika delar av planet då luften till exempel drar planet och skapar motstånd (formmotstånd), samt att vingytan skapar friktion som skapar drag, samt en inducerad drag som uppstår när vingarna producerar lyft. Storleken på dragkraften ges av

Där C\_D är den dimensionslösa dragkoefficienten.

Thrust är den framdrivande kraft som alstras av motorer eller propellrar, riktad framåt för att övervinna draget som är den motverkande kraften. Propellrar accelererar luft bakåt varvid en lika stor, framåtriktad reaktionskraft genereras, medan jetmotorer utvecklar drag genom att suga in luft, komprimera och avge heta avgaser med hög hastighet.

I stabil rak flygning är lyftkraften lika med tyngdkraften och kraften som drar flygplanet (thurst) lika med drag/motstånd, vilket gör att alla krafter tar ut varandra och planet åker i konstant hastighet. Att förstå samspelet mellan dessa fyra krafter är grundläggande för utformningen av en RL baserad autopilot.

### Manövrerbarhet, kontrollytor och styrprinciper

Kontrollytor och manövrerbarhet

* Höjdroder: Enligt Wikipedia (Mars 2024) så är höjdroder en rörlig yta på den horisontella stabilisatorn som reglerar planets pitch genom att förändra vinkeln mot luftströmmen. Utslag uppåt sänker nosen och utslag nedåt höjer nosen, vilket direkt påverkar anfallsvinkel och lyft.
* Skevroder: Enligt Wikipedia (April 2025) sitter skevroder på vingarnas bakkant och rör sig i motsatta riktningar för att skapa differential lyft, vilket ger ett vridmoment kring längdaxeln och inducerar roll (bank).
* Sidroder: Enligt Skybrary (u. å.) är sidrodret monterad på vertikala stjärtpartiet och kontrollerar yaw genom att skapa sidokrafter. Detta används främst för att motverka skevroderbroms (adverse yaw).

Skybrary (u. å.) nämner att när skevrodret vrider planet i roll ser man först en nosrörelse mot svängriktingen (skevroderbroms). Detta är eftersom den övre vingen får mindre drag än den undre vingen, Detta kompenseras sedan med sidrodertillslag.

Enligt FlightGear Wiki (juli 2016) är det viktigt att belysa propellerns vridmoment. Propellerns vridmoment är ett rotationsmoment som uppstår när motorns effektomvandling vill vrida flygkroppen i motsatt riktning mot propellerns rotation. Detta vridmoment måste piloten aktivt motverka.

## Reinforcement Learning (RL)

### Markov Decision Processes

Enligt geeksforgeeks (Feb 2025) är ett Markov Decision Process (MDP) en formell modell för sekventiellt beslutsfattande där varje tillstånd ***s*** är tillräckligt för att beskriva framtida utvecklingen, det vill säga har Markovegenskapen. En MDP definieras av:

* Tillståndsrum ***S***, som är alla möjliga situationer agenten kan befinna sig i.
* Actions ***A***, som ärhandlingar agenten kan utföra i varje tillstånd.
* Övergångsfunktion , sannolikheten att gå från tillstånd ***s*** till ***s’*** då action ***a*** utförs.
* Belöningssignal , den belöning agenten får då den utför ***a*** i ***s***.
* Diskonteringsfaktor , som bestämmer hur framtida belöningar värderas i förhållande till omedelbara.

I praktiken strävar en RL-agent efter att hitta en policy som är en regel för val av handling varje tillstånd, som maximerar den förväntade, kumulativa belöningen. Genom upprepade interaktioner med miljön och uppdateringar enligt t.ex. Q-learning eller policygradientmetoder lär sig agenten successivt en allt bättre approximation av den optimala policyn.

### Value och policy baserade algoritmer

I RL skiljer man ofta mellan value baserad och policy baserade metoder beroende på hur de representerar och optimerar agentens beteende. Value baserade metoder lär en uppskattning av framtida kumulativa belöningar för varje tillstånd ***s*** samt action ***a*** par, och härleder sedan en policy genom att välja de actions som maximerar detta värde. Policy baserade metoder optimerar i stället direkt en parametiserad policy, vilket gör dem väl lämpade för problem med actionutrymmen där kontinuerliga värden används. Nedan beskrivs de viktigaste algoritmerna inom respektive kategori, samt förbättrade och utvecklade varianter som Twin Delayed DDPG (TD3).

#### VALUE BASERADE METODER

Enligt geeksforgeeks (Feb 2025) Q-learning är en modellfri, off policyalgoritm som bygger på att iterativt uppdatera en tabelluppslagning av värdefunktionen ,vilken försöker förutsäga den förväntade kumulativa belöningen för att utföra action ***a*** i tillstånd ***s*** och därefter följa den bästa policyn. Vid varje steg uppdateras enligt Bellman ekvationen med hjälp av inlärningshastighet och diskonteringsfaktor. Q-Learning kan hantera slumpmässiga övergångar och belöningar utan att känna till miljöns inre dynamik.

Deep Q-Network (DQN) kombinerar Q-Learning med djupa neurala nätverk för att approximera i stora tillståndsrum med kontinuerliga värden där en tabelluppslagning är omöjlig. DQN introducerade dessutom replay buffer och target network för att stabilisera inlärningen, vilket möjliggjorde framgångsrika tillämpningar på hög dimensionella problem som spel och bildtolkning.

#### POLICY BASERADE METODER

Enligt wikipedia (April 2025) Policy gradient metoder optimerar direkt parametrarna 𝜽 i en policy genom att följa gradienten av den förväntade kumulativa belöningen med avseende på 𝜽 (stochastic gradient ascent). Dessa metoder är naturligt lämpade för kontinuerliga actionsutrymmen men kan ge hög varians i sina uppdateringar.

Enligt Amaresh Marekar (Jun 2022) är Deep Deterministic Policy Gradient (DDPG) är en off policy actor critic metod som kombinerar deterministic policygradients med en kritiker som lär en Q-funktion, och en aktör som lär en deterministisk policy. DDPG använder replay buffer och target networks för att stabilisera inlärningen i kontinuerliga actionsutrymmen.

Enligt Wikipedia (April 2025) är Proximal Policy Optimization (PPO) är en on policygradient algoritm som förbättrar stabiliteten genom att begränsa hur mycket policyn får ändras i varje uppdateringssteg via en klippningsfunktion i det objektiv som optimeras. PPO har blivit standard inom deep RL tack vare enkel implementering och robusta prestanda.

Enligt Mathworks (u. å.) är Twin Delayed DDPG (TD3) bygger vidare på DDPG för att motverka en vanlig felkälla, vilket är överestimering av Q-värden. TD3 använder clipped double-Q learning, två kritikernätverk och fördröjda policyuppdateringar för att minska bias och förbättra stabilitet i kontinuerliga actionutrymmen. Dessa innovationer gör TD3 mer robust mot hyperparameter variationer och överträffar ofta DDPG på benchmarks.

### Reward funktion och straff

#### Grundläggande belöningsfunktion

Enligt Amit Yadav (sep 2024) använder man belöningsfunktionen för att kartlägga varje tillstånd, action par till en numerisk belöning som agenten erhåller omedelbart efter att ha utfört action ***a*** i tillstånd ***s***. Belöningen fungerar som agentens kompass och definierar uppgiften genom att göra vissa beteenden mer attraktiva än andra. Utan en väldefinierade reward skulle agenten sakna mål och inte kunna skilja mellan bra och dåliga handlingar.

#### Reward Shaping

Enligt Tim Miller (u. å.) är Reward shaping tekniken att lägga till extra, mellanliggande belöningar för att ge agenten vägledning mot målet. När shaping är en potential baserad garanteras att optimal policy inte förendras. Man definiearar en potentialfunktion och lägger till belöningen

så att inlärningen försnabbas men det optimala handlingsmönstret förblir detsamma.

#### Straff och negativa belöningar

För att avskräcka oönskade eller farliga handlingar används negativa belöningar eller straff. Dessa kan vara proportionella mot graden av överträdelse, till exempel storlek på kursavvikelse, överbelastning av kontrollytor eller energiförbrukning. Att inkludera straff hjälper agenten att snabbt lära sig undvika ineffektiva eller riskfyllda manövrar och minskar risken för att fastna i suboptimala beteenden.

### Mätetal för prestanda

För att objektivt kunna jämföra och utvärdera olika RL-modeller på rak kursstyrning behöver evaluera både inlärningens framgång samt styrningens kvalitet. Nedan beskrivs de vanligaste metrikerna och motivationen till varför just dessa används.

#### Kumulativ reward

* Vad det mäter: Summan av diskonterade belöningar över en hel episod eller evaluaringsperiod
* Varför den används: ger en övergripande indikation på hur väl agenten maximerar det mål vi definierat via belöningsfunktionen, det vill säga hur bra den balanserar kort och långsiktiga mål under hela flygningen. En stigande ***G*** över träningsperioden indikerar att agenten lär sig.

#### Root Mean Squared Error (RMSE)

* Vad mäter det: Den genomsnittliga kvadrerade avvikelsen mellan startkursen **k aktuell** och vår faktiska kurs vilket är **k start**
* Varför vi använder det: RMSE ger ett tydligt, skalberoende mått på precisionen i styrningen där större fel väger tyngre. För flygautopiloten är det avgörande att hålla kursavvikelsen så låg som möjligt, och RMSE är ett standardiserat sätt att mäta detta.

#### Overshoot

* Vad det mäter: Tiden från en störning eller kursförändring tills att kursavvikelsen förblir inom ett fördefinierat felspann runt målvärdet.
* Varför vi använder det: Det visar hur snabbt vår agent kan korrigera avvikelser och återgå till rak kurs. Kort stabiliseringstid är viktig för att minimera tiden utanför önskat kursläge.

Denna kombination ger en heltäckande bild av hur väl en RL-agent lär sig uppgiften, men också hur väl den presterar i termer av noggrannhet, stabilitet och snabbhet, vilket är väldigt kritiska egenskaper för en flygautopilot.

# Metod

I detta kapitel beskrivs hur experimentmiljön byggdes upp, vilka verktyg och bibliotek som användes samt hur träning och utvärdering av RL-agenten genomfördes.

## Miljöintegration med MSFS 2024

För att skapa en realistisk träningsmiljö integrerades Microsoft Flight Simulator 2024 (MSFS 2024) med RL-agenten via en egenutvecklad socket-lösning. Denna socketintegrering skapades med C++ för att kunna skicka miljödata från MSFS till agenten. Detta gör att agenten kan ta del i miljön och styra flygplanet i MSFS och samtidigt få nödvändig data från simulatorn för att träna och nå vårt mål. MSFS 2024 erbjuder en högupplöst fysiksimulering av flygplansdynamik och atmosfäriska förhållanden, vilket gör det väl lämpat för att träna autopilotalgoritmer under nära verklighetsförhållanden.

Socket servern använder sig av SimConnect API vilket är MSFS officiella kommunikationsgränssnitt för att ta ut data från simulatorn. Sedan implementerades en TCP/IP-baserad server som kontinuerligt skickar utvald data tagen från simulatorn till RL-agenten. Agenten som är skriven i Python lyssnar på statusmeddelanden från MSFS och svarar med kontrollkommandon som höjdroderryck och skevroderryck via samma TCP-kanal.

## RL-ramverk och modellimplementation

Som ramverk for Reinforcement Learning användes Stable-Baselines3, ett väl underhållet Python-bibliotek med implementationer av en rad populära algoritmer. Valet föll på detta bibliotek tack vare dess enkelhet att implementera algoritmer som TD3 som användes för att skapa denna modell, möjlighet till GPU-acceleration, samt stöd för callback funktioner som ger flexibilitet att logga och spara data och modeller.

### Definiering av observation, action-space och rewards

För att agenten snabbt och stabilt skulle kunna lära sig hålla rak kurs definierades observationen av miljön vid varje tidssteg som en vektor bestående av planets aktuella pitch, bank och heading, samt deras med deras första tidsderivator (dvs. vinkelhastighet). Pitch och bank kan båda anta värden mellan -180 till 180 grader medan heading består av riktningen till True north och kan anta värdena 0 till 359 grader. Detta val motiverades av att både själva vinkellägena samt hur snabbt dessa förändras är centrala för att bedöma stabilitet och förutse framtida rörelser. Agentens action-space bestod av två kontinuerliga kontrollsignaler, utslag på höjdroder för höjd/pitch reglering och skevroder för bankreglering. Sidroder valdes att inte tas med då den inte är riktigt nödvändig under rak kurs i flygning, utan är mer viktig vid andra procedurer inom flygning.

För att undvika att neurala nätverkets vikter hamnade i olika skalor och därigenom underlätta inlärningen normaliserades samtliga observationer och actions till intervallet [-1, 1] via linjär skalning utifrån realistiska extrema värden för varje variabel. För pitch och bank var dessa extrema värden mellan -30 och 30 grader. Detta är för att minska agentens möjlighet att svänga flygplanet förmycket och minimera fönstret som den verkar i. Headings extremvärden ligger mellan -10 och 10 grader då vi vill skapa ett ännu mindre fönster för kursen då målet är att ligga i rak kurs. Agentens kontrollsignaler som ligger mellan [-1, 1] bestämmer hur planet ska styras. Negativa värden styr höjdrodret uppåt och skevrodret åt höger och positiva värden gör motsatsen.

### Reward

Belöningsfunktioner konstruerades för att både uppmuntra rak kurs och begränsa överdrivna styrutslag. Dessa beskrevs enligt följande.

#### Håll Kurs belöningar

Dessa belöningar är till för att ge agenten en belöning när pitch () och bank () är nära noll grader, samt när summan av heading () före och nuvarande är nära noll vilket ger agenten en tydlig riktning på vad den bör fokusera på. är en egen konstant för att minska värdet där agenten kan få belöningar.

#### Vinkelhastighetsstraff

Denna bestraffning är till för att ge agenten ett straff när vinkelhastigheterna inte är 0. Detta är för att få agenten att inte svänga eller avvika från sin kurs. är en vikt för att reglera hur viktig denna bestraffning är för agenten att följa.

#### Begränsning av kontrollutslag

Denna belöning är för att uppmuntra agenten att hålla kontrollytornas utslag små och undvika onödigt stora roderrörelser. Detta är för att ge en mer mjukare styrning av flygplanet. är agentens normaliserade utslag på kontrollkomponenten ***i*** (höjdroder respektive skevroder) vid tidpunkt ***t.*** är tröskelvärdet i normaliserade enheter, när ger , annars **.** är en vikt som bestämmer hur mycket denna bonus bidrar till den totala belöningen.

#### Mjukande av kontrollutslag

Denna bestraffning är till för att hålla agenten från att utföra för stora roderutslag och begränsa den till en tröskel som inte överstiger . När agenten överstiger denna tröskel så får den ett straff. är den euklidiksa normen action-vektorn , det vill säga den totala styrinsatsen vid tidpunkt ***t***. är en vikt som styr hur starkt agenten straffas för plötsliga eller stora kontrollutslag.

#### Säkerhetsstraff

Denna bestraffning säkerställer att agenten håller flygplanet inom en säker manöverzon genom att omedelbart bestraffa agenten när den ligger utanför denna säkerhetszon. och är agentens normaliserade pitch, respektive bank vinkel vid tidpunkt ***t.*** är den definierade normaliserade säkerhetströskelns som motsvarar maximalt tillåten lutning eller stigning. är en hög straffkostnad som avskräcker agenten från att gå utanför det säkra flygområdet.

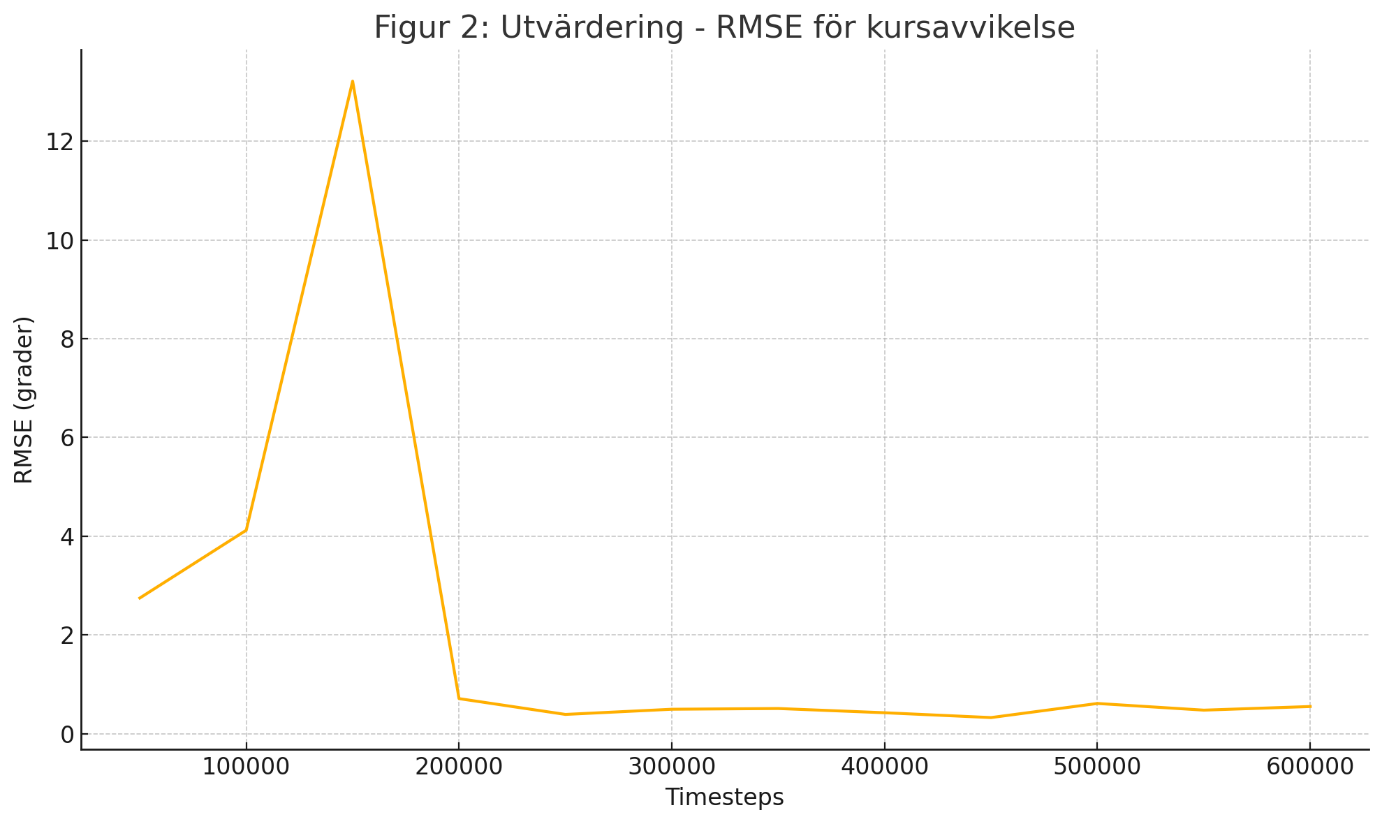
Agenten fick träna i varje episod i en simuleringsmiljö som startar flygplanet på 2000 m höjd, med en konstant indikerad hastighet på 150 knop, på en initial heading på 270 och en vind på 2 knop från samma riktning. Planet placerades i rak kurs i början av varje episod. Varje tränings eller utvärderingsavsnitt varade cirka 10 sekunder varefter miljön återställdes till sin startposition för att agenten ska få generalisera över flera initialtillstånd. Under evaluering testat agenten 20 gånger i rad efter varje 50 000 timesteps och medelvärdet från dessa evalueringar tas för att bestämma hur den testperioden gick.

# Resultat och Diskussion

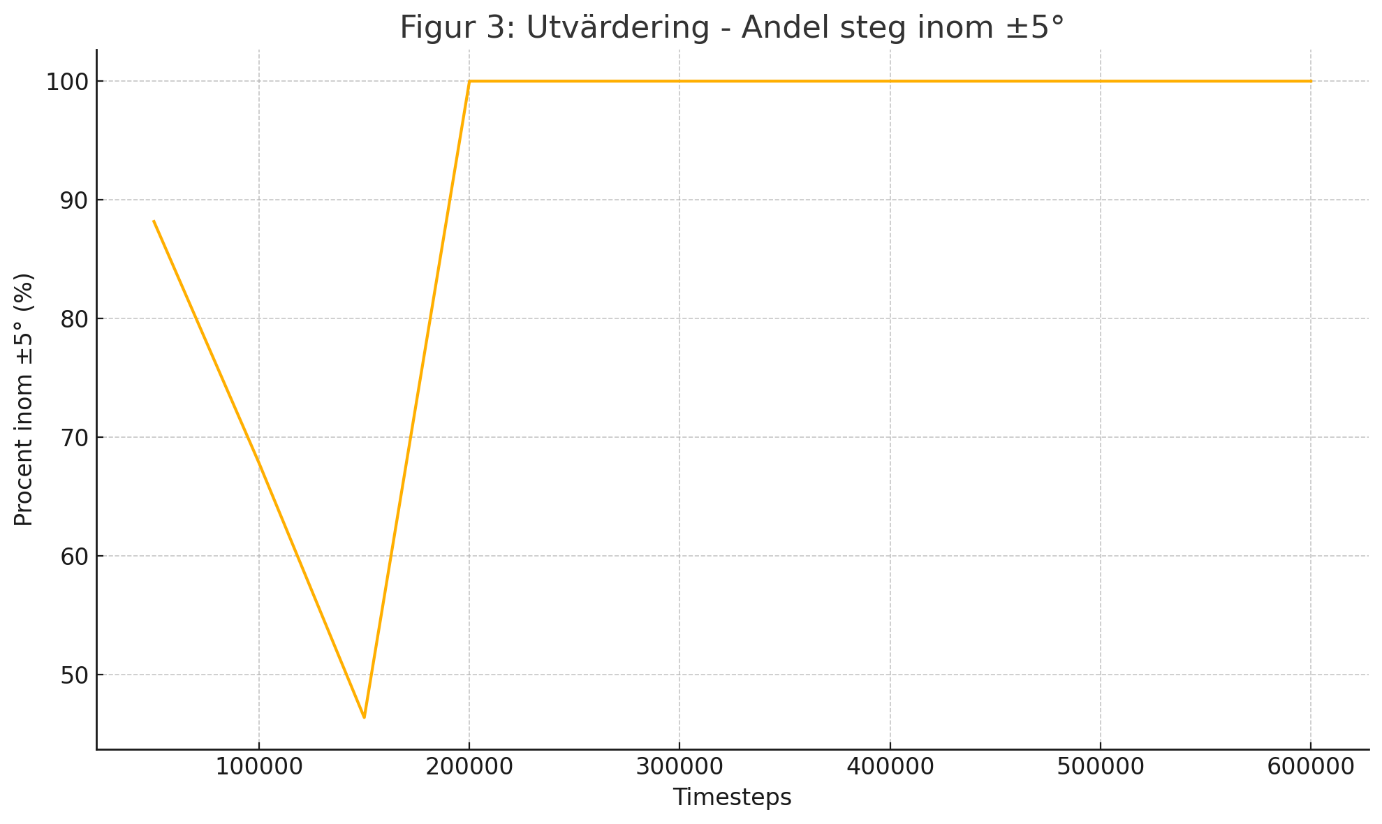
I det här avsnittet presenteras de viktigaste utvärderingsmåtten för den inlärda autopiloten, vilket är kumulativ reward, RMSE, overshot, stabiliseringstid och kontrollinsats. Resultaten kommer redovisas med tidsseriegrafer över tränings och utvärderingsförlopp.

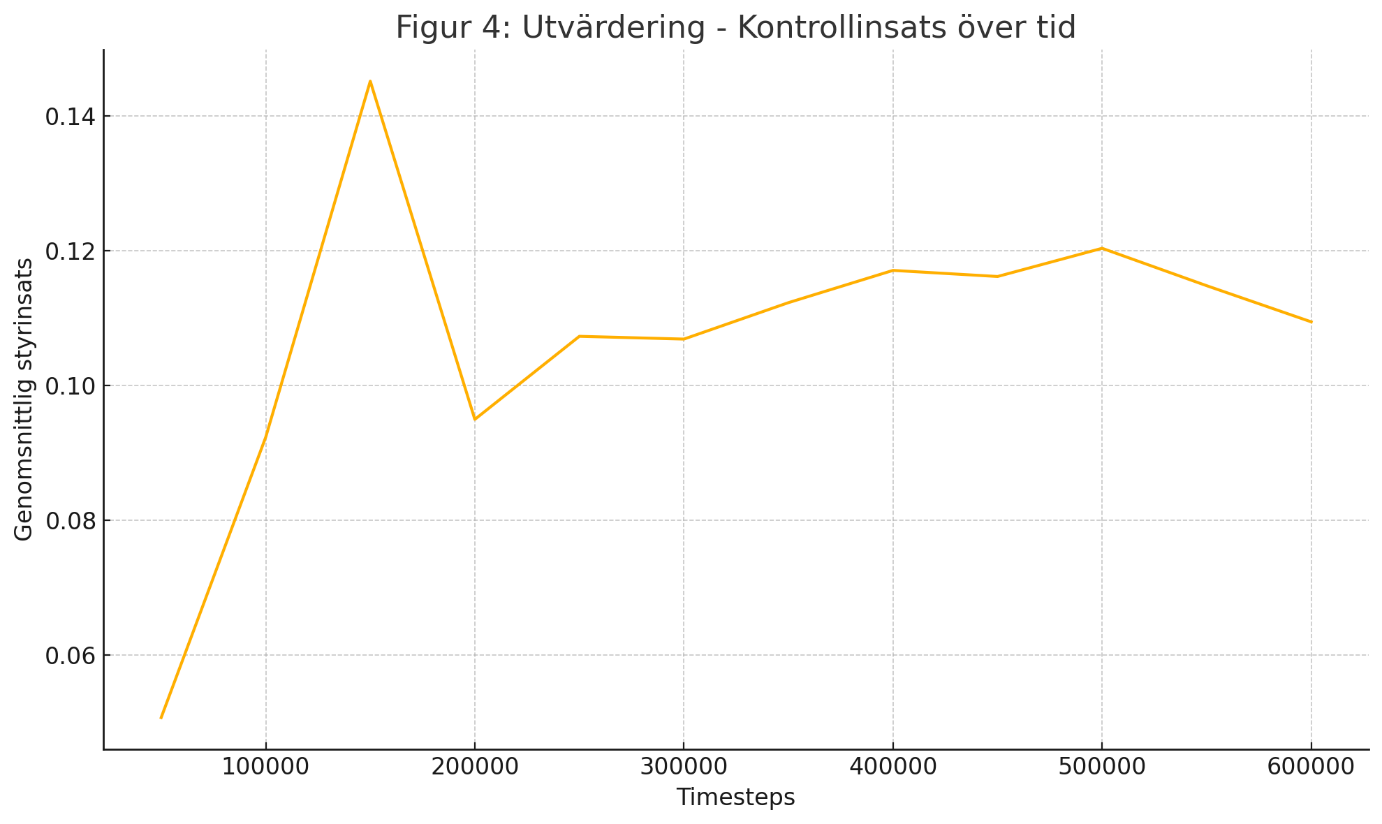
A graph with orange line

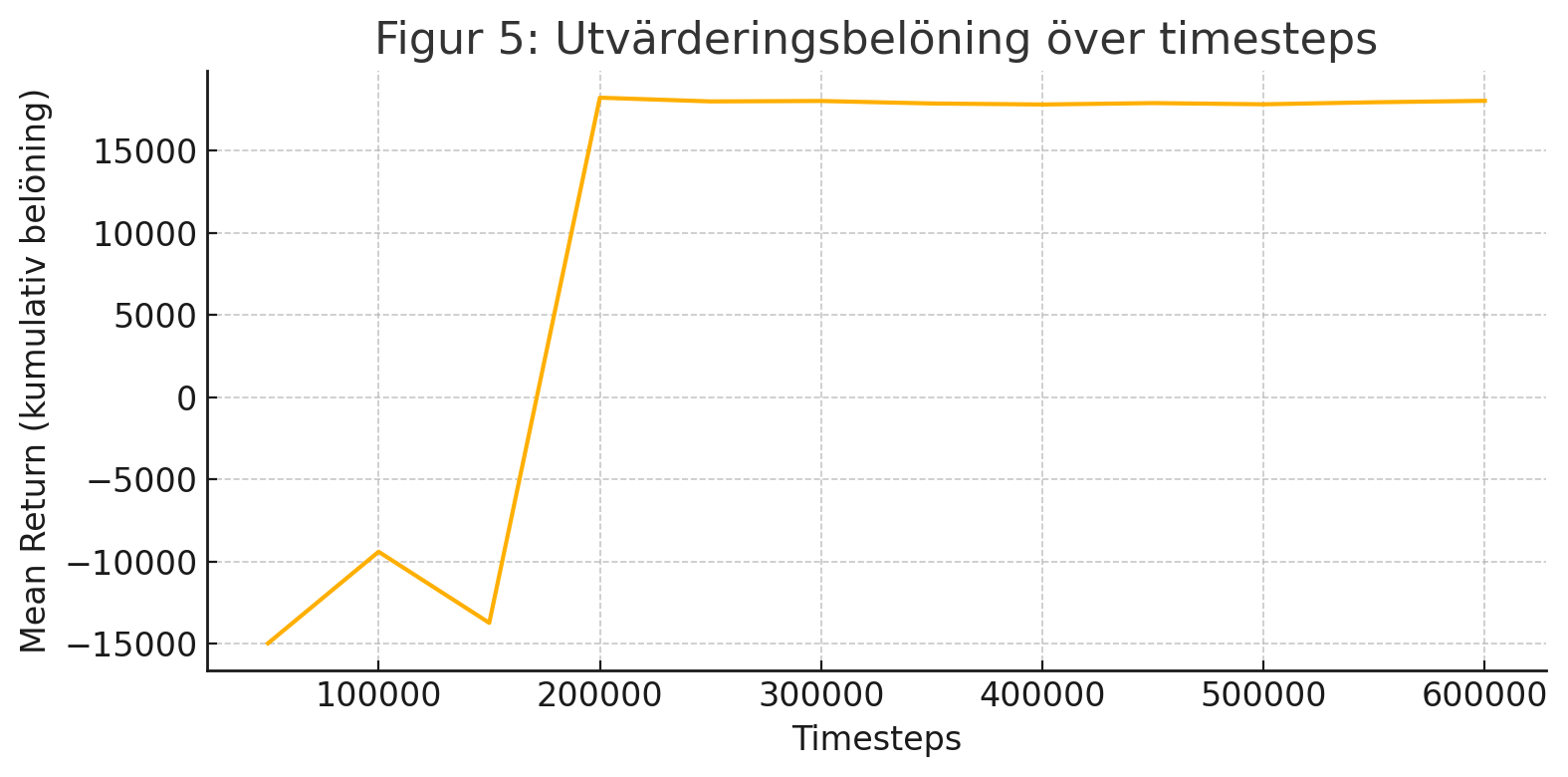
AI-generated content may be incorrect. Figur 1: Träningsprogress – Reward vs Timesteps.



Figur 2: Utvärdering – RMSE för kursavikelse

 Figur 3: Utvärdering – Andel steg inom +-5 grader

 Figur 4: Utvärdering – kontrollinsats över tid

 Figur 5: Utvärdering – Utvärderingsbelöning över timesteps.

|  |  |
| --- | --- |
| **Slutliga Utvärderingsmått** | |
| Medel utvärderings belöning | 18047 |
| RMSE | 0,55 |
| Procent inom 5 grader | 100.0 |
| Medel agentutslag | 0,109 |

Tabell 1: Slutliga Utvärderingsmått

Figur1 visar agentens genomsnittliga episodbelöning under träningen över timesteps. I början ligger belöningen runt -4000 men senare efter cirka 150 000 timesteps går agenten över nollpunkten och stiger snabbt i sin reward. Precis innan 300 000 timesteps konvergerar agenten kring 3 000, vilket tyder på att agenten har lärt sig en stabil och effektiv styrpolicy för rak kurs.

Figur 2 visar hur kursavvikelsens RMSE utvecklas under utvärderings­körningarna. Till en början ligger felet kring 3°–4°, stiger till en topp på ~13° vid 150 000 timesteps (utforskningsfas), för att sedan sjunka dramatiskt till under 1° vid cirka 200 000 timesteps. Därefter stabiliserar sig RMSE runt 0,4°–0,6°, vilket indikerar att agenten konsekvent håller kursen med hög precision.

Figur 3 visar andelen tidssteg där kursavvikelsen ligger inom ±5° under utvärderingsperioden. Till en början klarar agenten cirka 88 % vid 50 000 timesteps, men prestandan sjunker till omkring 46 % runt 150 000 timesteps under utforskningsfasen. Efter cirka 200 000 timesteps når agenten 100 % korrekthet inom toleransen och bibehåller detta genom hela den fortsatta inlärningen, vilket indikerar en fullt stabiliserad kurskontroll.

Figur 4 visar hur den genomsnittliga styrinsatsen (mean effort) utvecklas under utvärderingsfasen. I början ligger den runt 0,05, stiger till en topp på cirka 0,15 under utforskningsfasen vid ~150 000 timesteps, faller därefter till ~0,095 vid 200 000 steps och stabiliserar sig sedan runt 0,11–0,12. Detta tyder på att agenten till en början kräver större roderutslag för att utforska, men sedan konvergerar mot en mer energieffektiv och jämn styrstrategi.

Figur 5 visar hur den kumulativa belöningen under utvärderingskörningar (mean return) utvecklas med ökat antal timesteps. Efter omkring 200 000 steps stabiliserar sig belöningen kring 18 000, vilket indikerar att agenten uppnår maximal kurshållningsförmåga utan ytterligare inlärningsförbättring.

# Slutsatser och Diskussion

## Slutsats

Utifrån våra resultat kan vi med hög säkerhet svara på frågeställningen:

1. Kan en reinforcement learning-modell styra ett flygplan i rak kurs med en deviation inom 5 grader i roll och pitch?

Våra experiment visar att modellen stabiliseras kring en riktad flygning utan att avvika mer än +-5 grader från rak kurs. Detta framgår av Figur 1 och Figur 5. Vi kan se på dessa grafer att inlärningen har konvergerat till en platå där ytterligare träning inte ger en förbättrad kursstabilitet. I de övriga graferna och tabellen bekräftas att agenten håller sin kurs efter konvergens, vilket indikerar på att målet för en effektiv autopilot har nåtts. Modellen överträffar därmed förväntningarna vad gäller kursnoggrannhet och kursstabilitet.

Sammantaget visar studien att en vältränad RL-modell kan uppnå och bibehålla en stabil rak kurs inom de specificerade toleranserna, vilket besvarar det centrala syftet med denna rapport.

## Begränsningar och vidareutveckling

Ett första förbättringsområde är miljömodellen. I våra experiment har agenten tränats och utvärderats under mycket stabila, idealiserade förhållanden. Det är därför oklart hur modellen presterar under tuffare väder såsom storm, turbulenta vindkast eller regn, där dynamiken förändras snabbt och sensormätningar blir brusigare. För framtida studier rekommenderas att införa realistisk vädersimulering och spontant ge den olika väder miljöer för att öka agentens robusthet.

Ett andra område gäller agentens styrning. I den nuvarande implementationen hanterar agenten endast höjd- och skevroder, medan gasregleringen sköts manuellt eller inte alls. Detta leder till att agenten ibland måste kompensera med nedåtriktade roderutslag för att motverka överdriven stighastighet, vilket ökar kontrollbördan och kan orsaka onödiga kontrollryckningar. För att undvika detta bör gasregleringen integreras i träningsmiljön med antingen som en extra output från RL‑modellen eller som en separat reglerkrets, så att agenten kan optimera både kursstabilitet och kontroll av flygplanets hastighet medan det flyger.

# Appendix A

https://drive.google.com/file/d/1RCoV2nTDnZWZo9qo0PKu1jVgsRtC5ONj/view?usp=sharing

*Appendix A.1: Ingen kontroll av roder*

https://drive.google.com/file/d/1t6HA8Yxx1Mq00I5v\_7KqZOriwBrH0l-K/view?usp=sharing

Appendix A.2: Första episoderna

https://drive.google.com/file/d/1aIqokKocqac5I3YC1aaab4siPDW4YapG/view?usp=sharing

*Appendix A.3: Slutgiltiga modellen*

De olika filmklippen visar hur agenten styr flygplanet underträningens gång, vilket man kan se i Appendix A.2. I Appendix A.1 kan man se hur flygplanet agerar vid noll roderutslag, och i Appendix A.3 kan man se hur agenten kontrollerar flygplanet efter träningen, allt i samma miljö.

All kod som användes för att utföra detta projekt kan hittas på github: [THEXBOXNERD81/autopilot](https://github.com/THEXBOXNERD81/autopilot). Samt kan alla tabeller hittas som använder för att skapa alla grafer i rapporten. I koden finns alla värden som användes för att justera modellen och skapa den modell där dess tränings och evaluering data, som förekommer i rapporten.

# Källförteckning

Studyflight.com (u. å.). Understanding aerodynamic forces in flight (lift, weight, thrust, drag). Hämtad 2 maj 2025, från https://www.studyflight.com/understanding-the-aerodynamic-forces-in-flight/

Wikipedia (Mars 2024). Elevaror (aeronautics). https://en.wikipedia.org/wiki/Elevator\_%28aeronautics%29

Wikipedia (April 2025). Aileron. https://en.wikipedia.org/wiki/Aileron

Skybrary (u. å.). Rudder. Hämtad 2 maj 2025, https://skybrary.aero/articles/rudder

Flight wiki (April 2016). Understanding Propeller Torque and P-Factor. https://wiki.flightgear.org/Understanding\_Propeller\_Torque\_and\_P-Factor

GeeksforGeeks (Feb 2025). Markov Decision Process (MDP) in Reinforcement Learning.

https://www.geeksforgeeks.org/what-is-markov-decision-process-mdp-and-its-relevance-to-reinforcement-learning

GeeksforGeeks (Feb 2025). Q-Learning in Reinforcement Learning. <https://www.geeksforgeeks.org/q-learning-in-python>

MATLAB (u.å.) Twin-Delayed Deep Deterministic (TD3) Policy Gradient Agent. Hämtad 2 maj 2025, [https://se.mathworks.com/help/reinforcement-learning/ug/td3-agents.html](https://se.mathworks.com/help/reinforcement-learning/ug/td3-agents.html?utm_source=chatgpt.com)

Amaresh Marekar (Jun 2022). How DDPG (Deep Deterministic Policy Gradient) Algorithms works in reinforcement learning?. <https://medium.com/%40amaresh.dm/how-ddpg-deep-deterministic-policy-gradient-algorithms-works-in-reinforcement-learning-117e6a932e68>

Wikipedia (April 2025). Policy gradient method. [https://en.wikipedia.org/wiki/Policy\_gradient\_method](https://en.wikipedia.org/wiki/Policy_gradient_method?utm_source=chatgpt.com)

Wikipedia (April 2025). Proximal policy optimization. https://en.wikipedia.org/wiki/Proximal\_policy\_optimization

Amit Yadav (Sep 2024). Reward Function in Reinforcement Learning. <https://medium.com/biased-algorithms/reward-function-in-reinforcement-learning-c9ee04cabe7d>

Tim Miller, The University of Queensland (u. å.) Reward shaping. Hämtad 2 maj 2025, https://gibberblot.github.io/rl-notes/single-agent/reward-shaping.html