CHALMERS

Att följa ideala trajektorier med kinodynamiskt begränsade robotar

En evaluering av LQR och MPC med och utan CBF.

Examensarbete, kurskod EENX20 vid avdelningen för System- och Reglerteknik på Institutionen för Elektroteknik

AHMED YAMAN ALKHUZAEE
EYAD TAHHAN

INSTITUTIONEN FÖR ELEKTROTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2026
www.chalmers.se

Examensarbete, kurskod EENX20 2026

Att följa ideala trajektorier med kinodynamiskt begränsade robotar

En evaluering av LQR och MPC med och utan CBF.

AHMED YAMAN ALKHUZAEE
EYAD TAHHAN

CHALMERS

Institutionen för Elektroteknik
System- och Reglerteknik
Forskargruppen för Automation
Chalmers tekniska högskola
Göteborg, Sverige 2026

Att följa ideala trajektorier med kinodynamiskt begränsade robotar
En evaluering av LQR och MPC med och utan CBF.
AHMED YAMAN ALKHUZAEE
EYAD TAHHAN

Handledare: Alvin Combrink
Biträdande Handledare: Yingshuai Quan
Examinator: Martin Fabian

Examensarbete, kurskod EENX20 2026
Institutionen för Elektroteknik
System- och Reglerteknik
Forskargruppen för Automation
Chalmers tekniska högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Att följa ideala trajektorier med kinodynamiskt begränsade robotar
En evaluering av LQR och MPC med och utan CBF.
AHMED YAMAN ALKHUZAEE
EYAD TAHHAN
Institutionen för Elektroteknik
Chalmers tekniska högskola

Förord

Detta examensarbete har genomförts vid Chalmers tekniska högskola inom institutionen för Elektroteknik. Arbetet har haft fokus på reglering och simulering av flera robotar, där målet har varit att undersöka hur ideala rörelseplaner kan följas av robotar med verkliga begränsningar i rörelse och styrning.

Vi vill rikta ett stort tack till vår handledare Alvin Combrink för stöd, vägledning och värdefulla synpunkter under arbetets gång. Vi vill även tacka vår biträdande handledare Yingshuai Quan för teknisk hjälp och goda råd genom projektet. Slutligen vill vi tacka vår examinator Martin Fabian samt alla som på något sätt har bidragit till arbetet.

Ahmed Yaman Alkhuzaee och Eyad Tahhan, Göteborg 2026

Abstract

Autonomous mobile robots are becoming increasingly common in production, warehouses, and other automated environments. When several robots move in the same environment, their motions must be coordinated to avoid collisions. Planning collision free motions is a well known and difficult problem, and simplified motion models are often used to make the planning easier. However, these simplifications can make it difficult for real robots to follow the planned paths exactly, which can reintroduce the risk of collisions.

This thesis investigates how well robots with more constrained, and therefore possibly more realistic, motion models can follow plans created using simplified motion assumptions. The robots are described using a kinematic bicycle model, where the motion is limited by velocity, acceleration, and steering angle. In contrast, the planning model assumes omnidirectional motion with unlimited acceleration. Four control strategies are compared: Linear Quadratic Regulator (LQR), Model Predictive Control (MPC), LQR with Control Barrier Functions (LQR+CBF), and MPC with Control Barrier Functions (MPC+CBF).

The results show that LQR works in simpler scenarios, but has difficulties with sharp turns and when several robots move close to each other. MPC provides better plan tracking and more stable motion. When CBF is used, the number of collisions is reduced, but not all collisions are eliminated in the tested scenarios. Overall, MPC+CBF gives the best balance between plan tracking and collision avoidance.

Keywords: multi-robot simulation, path tracking, kinematic bicycle model, LQR, MPC, CBF, CasADi, JSON.

Sammanfattning

Autonoma mobila robotar blir allt vanligare i produktion, lager och andra automatiserade miljöer. När flera robotar rör sig i samma miljö så måste deras rörelser koordineras för att undvika kollisioner. Planeringen av dessa kollisionsfria rörelser är ett välkänt och svårlöst problem, vilket ofta underlättas genom att använda förenklade modeller över robotarnas rörelser. På grund av dessa förenklingar så kan det hända att robotarna inte kan följa planerna exakt, vilket återinför risk för kollisioner.

Detta arbete undersöker hur väl robotar med mer begränsade — och därav möjligen mer realistiska rörelsemodeller — kan följa planer som skapas under förenklade rörelsemodeller. Robotarna beskrivs med en kinematisk cykelmodell, där rörelsen begränsas av hastighet, acceleration och styrvinkel, medan modellen som används under planering antar omnidirektionell styrning med obegränsad acceleration. Fyra reglerstrategier jämförs: Linear Quadratic Regulator (LQR), Model Predictive Control (MPC), LQR med Control Barrier Functions (LQR+CBF), och MPC med Control Barrier Functions (MPC+CBF).

Resultaten visar att LQR fungerar i enklare scenarier, men får svårigheter vid skarpa svängar och när flera robotar rör sig nära varandra. MPC ger bättre planföljning och mer stabila rörelser. När CBF används minskar antalet kollisioner, men alla kollisioner elimineras inte i de testade scenarierna. Sammantaget ger MPC+CBF bäst balans mellan planföljning och kollisionsundvikning.

List of Acronyms

Below is the list of acronyms and technical terms that have been used throughout this thesis listed in alphabetical order:

CasADi	A software framework for numerical optimization
CBF	Control Barrier Function
IPOPT	Interior Point Optimizer
JSON	JavaScript Object Notation
LQR	Linear Quadratic Regulator
MAPF	Multi-Agent Path Finding
MPC	Model Predictive Control
QP	Quadratic Programming
RK4	Fourth-order Runge-Kutta method

Innehåll

Figurer

Tabeller

1

Inledning

1.1 Bakgrund

I takt med att automatiserade lager och autonoma robotsystem blir allt vanligare, ökar behovet av effektiva algoritmer för att koordinera flera robotar på en gemensam yta. Detta problemområde kallas för Multi-Agent Path Finding (MAPF), där målet är att planera rörelser för flera agenter utan att de kolliderar med varandra [1].

MAPF-problem löses ofta genom att generera planer i form av sekvenser av positioner och tidstämplar, till exempel $[(p_1, t_1), (p_2, t_2), \dots]$. I dessa planer antar man vanligtvis att agenterna är omnidirektionella, vilket innebär att de kan röra sig fritt i alla riktningar utan begränsningar. I praktiken ser det dock inte ut så.

De flesta verkliga robotar och fordon följer kinematiska begränsningar. En vanlig modell som används för att beskriva detta är den så kallade cykelmodellen, där roboten inte kan röra sig i sidled utan måste svänga för att ändra riktning, på samma sätt som en vanlig bil. En liknande kinematisk modell används av Shen och medförfattare. för Ackermann-styrda AMR robotar, där robotens position, riktning, hastighet och styrvinkel ingår i rörelsemodellen [2]. Detta gör att det uppstår ett gap mellan den teoretiska planeringen och hur roboten faktiskt kan röra sig i verkligheten. För att en robot ska kunna följa en sådan plan krävs därför ett reglersystem som tar hänsyn till dessa begränsningar. I detta arbete har två olika metoder undersökts för detta ändamål, den första är Linear Quadratic Regulator (LQR) [3] och den andra är Model Predictive Control (MPC) [4]. Arbetet började med att implementera en LQR-regulator för banföljning. LQR är en relativt enkel och snabb metod som fungerar bra för att stabilisera system kring en given bana. Samtidigt visade det sig att metoden har begränsningar, särskilt när systemet blir mer komplext eller när flera robotar ska samverka.

Därför vidareutvecklades lösningen till att använda MPC med hjälp av CasADi [4], [5]. Till skillnad från LQR kan MPC ta hänsyn till både systemets begränsningar och framtida tillstånd genom att optimera rörelsen över en tidshorisont. Detta gör metoden mer flexibel och bättre lämpad för mer realistiska scenarier, till exempel när flera robotar rör sig samtidigt och krockrisk uppstår.

För att ytterligare hantera säkerhetsaspekter såsom krockundvikning, undersöks även Control Barrier Functions (CBF) i detta arbete. CBF används som ett komplement till både LQR och MPC för att säkerställa att robotarna håller ett säkert avstånd till varandra och till hinder under rörelse.

1.2 Syfte

Syftet med projektet är att utveckla ett simuleringsverktyg i Python för att visualisera och analysera rörelseplanering för flera agenter. Fokus ligger på att utvärdera och jämföra två olika regulatorstrategier för spårningskontroll av en robot som följer en given geometrisk referensbana. Arbetet inleds med implementering av en linjärkvadratisk regulator (LQR), baserad på en linjäriserad felmodell. Därefter vidareutvecklas systemet genom implementering av en modellprediktiv regulator (MPC) med hjälp av optimeringsramverket CasADi, i syfte att undersöka förbättringar i prestanda och flexibilitet.
Utöver detta undersöks även Control Barrier Functions (CBF) som en metod för att säkerställa säkerhetskrav, såsom krockundvikning. CBF appliceras som ett komplement till både LQR och MPC för att analysera hur säkerhetsbegränsningar påverkar systemets beteende.

1.3 Avgränsningar

Avgränsningar För att göra problemet hanterbart har följande avgränsningar gjorts:

- Endast en kinematisk modell används, det vill säga att ingen dynamik, såsom krafter eller massa tas med.
- Arbetet är helt baserat på simulering.
- Sensorfel och brus beaktas inte.
- Krockhantering sker enbart genom säkerhetsavstånd och CBF, inte genom samordnad optimering mellan robotar.

1.4 Precisering av frågeställningen

Frågeställningarna som arbetet besvarar är:

- Hur kan ett reglersystem LQR respektive MPC designas för att minimera avvikelsen i både position och tid för en robot med cykelmodell?
- Hur påverkas planens validitet när robotarnas fysiska begränsningar, såsom maximal styrvinkel och acceleration, introduceras i en multi-agent-miljö?
- Vilken av de två valda reglermetoderna lämpar sig bäst för att hantera de skarpa svängar och hastighetsförändringar som krävs för att följa en MAPF-plan?
- Hur påverkar införandet av Control Barrier Functions (CBF) systemets säkerhet och beteende?

2

Teoretisk bakgrund

I detta kapitel presenteras de grundläggande matematiska modeller och reglersystem som ligger till grund för arbetet.

2.1 Kinematisk cykelmodell

För att beskriva en robots rörelse används den kinematiska cykelmodellen, som är nära kopplad till Ackermann styrning [2]. Modellen är vanligt förekommande inom robotik och autonoma fordon, eftersom den tar hänsyn till att systemet inte kan röra sig i sidled utan måste ändra riktning genom att svänga. Modellen antar att roboten har två hjulaxlar där styrningen sker via framhjulen. Rörelsen beskrivs med hjälp av position (x, y) samt orientering ψ. Dessa tillstånd påverkas av robotens hastighet v och styrvinkel δ.

Rörelsen beskrivs av följande samband:

$$\dot{x} = v \cos \psi \tag{2.1}$$

$$\dot{y} = v \sin \psi \tag{2.2}$$

$$\dot{\psi} = \frac{v}{L} \tan \delta \tag{2.3}$$

där L är avståndet mellan fram och bakaxel.

Denna modell används i arbetet som grund för både LQR och MPC-reglering, eftersom den ger en realistisk beskrivning av robotens rörelse samtidigt som den är relativt enkel att arbeta med [2].

2.2 Numerisk integration och Runge-Kutta 4

För att simulera robotens rörelse måste den kontinuerliga rörelsemodellen lösas numeriskt. Eftersom modellen beskriver hur tillstånden förändras över tid används en numerisk integrationsmetod för att beräkna robotens nästa tillstånd vid varje tidssteg.

I detta arbete används Runge-Kutta 4 (RK4) [6], som är en vanlig metod för numerisk integration av differentialekvationer. Metoden beräknar flera uppskattningar av derivatan inom samma tidssteg och kombinerar dessa för att få en mer noggrann approximation av nästa tillstånd än vad enkel Euler-integration ger.

RK4 är särskilt lämplig i detta arbete eftersom robotens rörelse påverkas av både hastighet och styrvinkel. När roboten svänger eller accelererar kan en enkel Eulermetod ge större fel, medan RK4 ger en mer stabil och noggrann uppdatering av robotens position, riktning och hastighet.

2.3 MAPF och rörelsemodeller

Inom området Multi Agent Path Finding (MAPF) genereras ofta lösningar där agenter antas vara omnidirektionella, vilket innebär att de kan röra sig fritt i alla riktningar. I praktiken är detta dock inte realistiskt, eftersom verkliga robotar är begränsade av sin kinematik. Detta skapar ett gap mellan teoretisk planering och faktisk rörelse [1].

För att göra planerade banor mer körbara för mobila robotar kan olika metoder användas, exempelvis banutjämning (path smoothing) [7] och reglersystem för att följa en referensbana [2].

I multirobotsystem är det dessutom viktigt att inte bara planera varje robots rörelse, utan även att samordna robotarna så att uppgifter kan fördelas och konflikter mellan agenter kan undvikas. Detta kopplar MAPF-problemet till bredare problem inom uppgiftsfördelning och koordinering i multirobotsystem [8], [9].

LQR är en etablerad metod inom optimal reglering och kan användas för stabilisering och referensföljning av dynamiska system [3]. Inom mobila robotsystem har rörelseplanering kombinerats med banföljningsreglering för flera autonoma mobila robotar med icke-holonomiska begränsningar [2]. MPC är en flexibel metod eftersom den kan ta hänsyn till begränsningar och framtida tillstånd, och den kan implementeras med hjälp av optimeringsverktyg som CasADi [4].

När det gäller säkerhet har Control Barrier Functions (CBF) blivit en viktig metod för att garantera kollisionsfri rörelse. CBF används ofta tillsammans med andra regulatorer för att säkerställa att roboten håller ett säkert avstånd till hinder och andra robotar [10], [11].

2.4 Reglertekniska principer

I detta arbete används tre huvudsakliga metoder: LQR, MPC och CBF.

2.4.1 Linear Quadratic Regulator (LQR)

Linear Quadratic Regulator (LQR) är en reglerstrategi för linjära system på formen

$$\dot{x} = Ax + Bu. \tag{2.4}$$

Metoden bygger på att minimera en kvadratisk kostnadsfunktion enligt

$$J = \sum \left(x^T Q x + u^T R u \right) \tag{2.5}$$

där Q och R är viktmatriser som bestämmer hur mycket tillståndsfel respektive styrinsats ska straffas.

LQR kallas optimal eftersom metoden ger den styrsignal som minimerar den valda kostnadsfunktionen för det linjära systemet. Med optimal menas alltså inte att metoden är bäst i alla situationer, utan att den är optimal med avseende på den givna modellen och de valda viktmatriserna Q och R.

LQR bygger på en linjär modell eller en linjärisering kring en arbetspunkt och kan inte hantera explicita begränsningar på exempelvis styrvinkel, acceleration och hastighet direkt i reglerproblemet. Detta innebär att LQR kan vara optimal i sin matematiska formulering, men ändå vara mindre lämplig i mer komplexa scenarier där roboten har olinjära rörelsebegränsningar, skarpa svängar eller där flera robotar rör sig nära varandra [3].

2.4.2 Model Predictive Control (MPC)

Model Predictive Control (MPC) [12] är en prediktiv reglerstrategi där ett optimeringsproblem löses vid varje tidssteg över en framtida tidshorisont N. Metoden använder en modell av systemet för att förutsäga framtida tillstånd och beräkna styrsignaler som minimerar en kostnadsfunktion.

I detta arbete används MPC för att minimera en kostnadsfunktion av typen:

$$J = \sum_{k=0}^{N} (x_k - x_{\text{ref}})^T Q (x_k - x_{\text{ref}}) + u_k^T R u_k \tag{2.6}$$

där x_k är det predikterade tillståndet vid tidssteg k, x_{ref} är referenstillståndet och u_k är styrsignalen. Matriserna Q och R används för att vikta referensfel respektive styrinsats.

Till skillnad från LQR kan MPC inkludera begränsningar direkt i optimeringsproblemet, exempelvis begränsningar på hastighet, acceleration och styrvinkel. Detta gör MPC mer beräkningskrävande än LQR, men också mer flexibel i scenarier där robotens fysiska begränsningar är viktiga.

MPC används därför i detta arbete som ett alternativ till LQR. Även om LQR är optimal med avseende på en linjär modell och en vald kvadratisk kostnadsfunktion, gäller detta under mer begränsade antaganden. Eftersom robotmodellen i detta arbete är olinjär och innehåller fysiska begränsningar är MPC bättre lämpad för situationer där systemet behöver planera flera steg framåt [4], [5].

2.4.3 Control Barrier Functions (CBF)

Control Barrier Functions (CBF) [10], [11] används för att säkerställa att systemet håller sig inom ett säkert område. I detta arbete betyder ett säkert område att roboten inte kommer för nära ett hinder eller en annan robot. Det säkra området definieras därför som alla positioner där avståndet till hindret eller den andra roboten är större än ett valt säkerhetsavstånd.

En vanlig barriärfunktion kan skrivas som:

$$h(x) = (x - x_{obs})^2 + (y - y_{obs})^2 - R_{safe}^2 \tag{2.7}$$

där x och y är robotens aktuella position, x_{obs} och y_{obs} är positionen för hindret eller den andra roboten, och R_{safe} är säkerhetsavståndet.

För att garantera säkerhet införs följande villkor:

$$h(x_{k+1}) - h(x_k) \geq -\alpha h(x_k) \tag{2.8}$$

där x_k är robotens tillstånd vid nuvarande tidssteg, x_{k+1} är robotens tillstånd vid nästa tidssteg och α är en positiv parameter som styr hur snabbt systemet får närma sig säkerhetsgränsen. Detta villkor säkerställer att roboten inte rör sig mot ett osäkert område på ett okontrollerat sätt [10], [13].

2.4.4 Sammanfattning av LQR, MPC och CBF

LQR och MPC är båda metoder för reglering, men de skiljer sig i hur de hanterar systemets komplexitet.

LQR är enkel och snabb, men bygger på en linjär approximation och saknar möjlighet att hantera begränsningar direkt [3]. MPC är mer avancerad och kräver mer beräkningskraft, men kan istället ta hänsyn till både framtida tillstånd och fysiska begränsningar [4].

I detta arbete används därför LQR som en grundläggande metod, medan MPC används för att förbättra systemets beteende i mer realistiska scenarier.

CBF används inte som en ersättning för LQR eller MPC, utan som ett komplement för att säkerställa säkerhet i systemet.

2.5 Feldefinition

Feldefinitionen beskriver hur robotens avvikelse från referensbanan beräknas och är en central del i både LQR och MPC-regleringen.

I detta arbete definieras felet i två delar: ett sidofel e_y och ett orienteringsfel e_ψ. Sidofelet beskriver avståndet mellan robotens position och referensbanan, medan orienteringsfelet beskriver skillnaden mellan robotens riktning och banans riktning.

Dessa fel kan uttryckas som:

$$e_y = -\sin(\psi_r)(x - x_r) + \cos(\psi_r)(y - y_r) \tag{2.9}$$

$$e_\psi = \psi - \psi_r \tag{2.10}$$

där (x_r, y_r) och ψ_r representerar referensbanans position och orientering.

Feldefinitionen används sedan som grund för att beräkna styrsignalen i regulatorerna. Genom att minimera dessa fel kan roboten följa referensbanan på ett stabilt sätt [2].

3

Metod

I detta kapitel beskrivs projektets tekniska genomförande, från val av mjukvaruverktyg till den arkitektoniska uppbyggnaden av kontrollsystemet.

3.1 Utvecklingsmiljö och verktyg

Simuleringen har utvecklats i programmeringsspråket Python, vilket valdes för dess omfattande bibliotek inom robotik, numerisk optimering och visualisering. De centrala verktygen som använts är:

- CasADi: Ett ramverk för numerisk optimering som används för att formulera och lösa de olinjära MPC och QP-problemen [4], [5].
- IPOPT: En mjukvara för storskalig olinjär optimering som fungerar som lösare för de problem som definieras i CasADi [14].
- NumPy: Används för numeriska beräkningar [15].
- Matplotlib: Används för att visualisera robotarnas rörelse samt generera statistiska grafer över systemets tillstånd och styrsignaler [16].

3.2 Reglerstrategi och experimentuppsättning

I simuleringen implementerades fyra reglerfall:

- LQR
- MPC
- LQR med CBF
- MPC med CBF

Utöver dessa reglerfall används även några viktiga delar i implementationen:

- RK4
- Robotklass
- CasADi
- QP

3.2.1 Implementering av LQR

LQR implementerades som en referensmetod för banföljning. Vid varje tidssteg beräknas felet mellan robotens aktuella tillstånd och referensbanan. Detta fel används för att bestämma styrsignalen genom en förstärkningsmatris K. Styrsignalen beräknas enligt:

$$\delta = \delta_r - Ke \tag{3.1}$$

där e består av sidofel och orienteringsfel [3].

I implementationen uppdateras hastigheten separat genom en enkel modell med acceleration och bromsning.

3.2.2 Implementering av MPC

Implementeringen av MPC bygger på att ett optimeringsproblem formuleras med tillstånd, styrsignaler och begränsningar över en prediktionshorisont, vilket sedan löses med hjälp av CasADi [4], [5].

Vid varje tidssteg löses ett optimeringsproblem över en horisont N, där tillstånd och styrsignaler optimeras för att följa referensbanan [12].

Tillståndsvariabler:

$$X = [x, y, \psi, v] \tag{3.2}$$

Styrsignaler:

$$U = [a, \delta] \tag{3.3}$$

Metoden tar hänsyn till systemets dynamik samt begränsningar på styrsignaler och tillstånd. Endast det första styrsteget från lösningen appliceras, vilket innebär att optimeringen upprepas vid varje tidssteg [12].

3.2.3 Implementering av CBF

Control Barrier Functions (CBF) implementerades som ett säkerhetslager för att minska risken för kollisioner mellan robotarna [10].

CBF används inte som en egen reglerstrategi, utan som ett tillägg till både LQR och MPC. Syftet är att säkerställa att roboten håller ett säkert avstånd till andra robotar.

I implementationen används ett QP-baserat filter som justerar styrsignalen vid behov [10]:

$$\min \|u^* - u\|^2 \tag{3.4}$$

där u är styrsignalen från LQR eller MPC och u^* är den justerade signalen. Säkerhetsvillkoret baseras på barriärfunktionen i (2.7) som definierades i teoridelen, samt i (2.8) används det för att säkerställa att roboten inte bryter mot säkerhetsavståndet. Detta innebär att:

- styrsignalen från LQR eller MPC används som grund, och
- CBF justerar endast signalen om säkerheten riskeras.

3.2.4 RK4

Runge-Kutta 4 (RK4) används för att beräkna robotens nästa tillstånd i simuleringen. Metoden ger en mer noggrann uppdatering än en enkel Euler-metod, eftersom den beräknar rörelsen i flera steg inom samma tidsintervall. Detta är viktigt när roboten svänger eller när hastigheten ändras.

3.2.5 Robotklass

En robotklass används för att hantera varje robot som ett eget objekt. Varje robot lagrar sitt tillstånd, sin referensbana, startpunkt, målpunkt och historik under simuleringen. Detta gör det enklare att simulera flera robotar samtidigt och jämföra deras rörelser.

3.2.6 CasADi

CasADi används för att formulera MPC-problemet. Med CasADi kan robotens modell, kostnadsfunktion och begränsningar skrivas som ett optimeringsproblem. Detta gör att MPC kan planera flera steg framåt och ta hänsyn till begränsningar som hastighet, acceleration och styrvinkel.

3.2.7 QP

Quadratic Programming (QP), används i CBF-delen som ett säkerhetsfilter. Först beräknar LQR eller MPC en styrsignal. Om det finns risk att robotarna kommer för nära varandra ändrar QP-filtret styrsignalen så lite som möjligt, men försöker ändå uppfylla säkerhetskravet [10].

3.3 Implementerad robotmodell

I simuleringen används en kinematisk cykelmodell för att beskriva robotarnas rörelse. Modellen valdes eftersom robotarna antas ha fordonsliknande rörelseegenskaper. Det innebär att roboten inte kan röra sig direkt i sidled, utan behöver röra sig framåt eller bakåt för att ändra riktning.

Robotens tillstånd beskrivs som:

$$\mathbf{x} = \Big[x, y, \psi, v\Big]^T \tag{3.5}$$

där x och y är robotens position, ψ är robotens riktning och v är hastigheten. Styrsignalerna är acceleration och styrvinkel:

$$u = \Big[a, \delta\Big]^T \tag{3.6}$$

Robotens tillstånd och styrsignaler illustreras i Figur 3.1. Accelerationen a används i simuleringen för att öka eller minska robotens hastighet, medan styrvinkeln δ används för att ändra robotens riktning. I modellen beskrivs roboten därför inte

endast som en punkt med position i planet, utan en kinematisk cykelmodell eftersom robotarna antas ha rörelsebegränsningar som liknar fordon, där riktning, hastighet och styrvinkel och acceleration påverkar hur roboten kan röra sig.

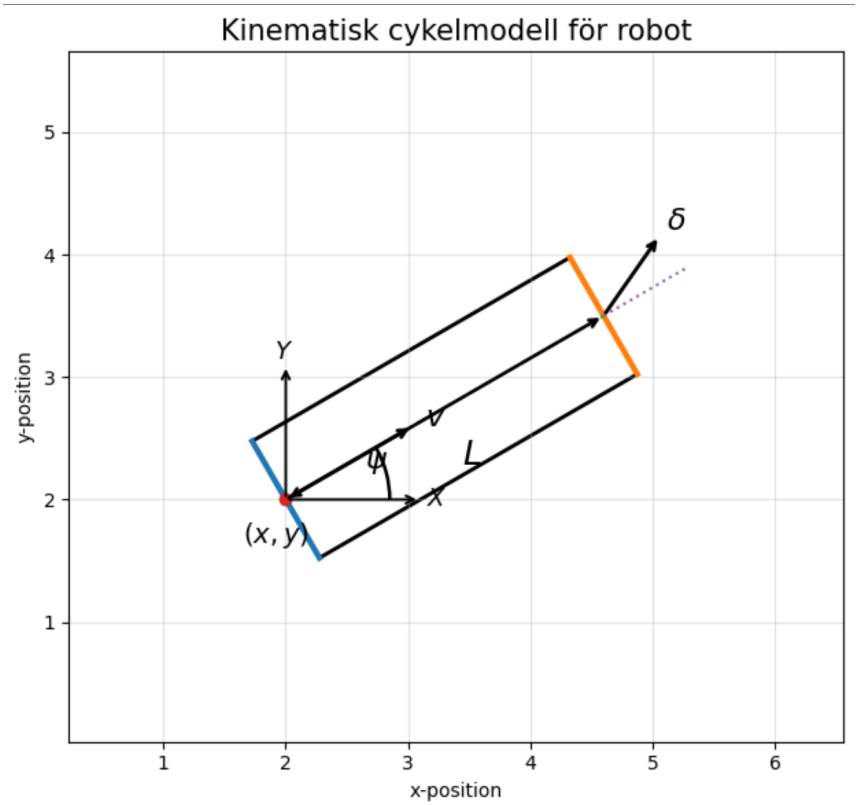
Robotens rörelse beskrivs med

$$\dot{x} = v\cos(\psi) \tag{3.7}$$

$$\dot{y} = v\sin(\psi) \tag{3.8}$$

$$\dot{\psi} = \frac{v}{L}\tan(\delta) \tag{3.9}$$

$$\dot{v} = a \tag{3.10}$$



Figur 3.1: Den kinematiska cykelmodellen som används i simuleringen (se [2]).

där L är robotens hjulbas [2]. I simuleringen begränsas både hastighet, acceleration och styrvinkel. Detta betyder att roboten inte kan svänga hur snabbt som helst och inte heller accelerera obegränsat. För att säkerställa fysiskt genomförbara rörelser begränsas därför robotens hastighet, acceleration och styrvinkel enligt

$$0 \leq v \leq v_{\max} \tag{3.11}$$

$$-a_{\max} \leq a \leq a_{\max} \tag{3.12}$$

$$-\delta_{\max} \leq \delta \leq \delta_{\max} \tag{3.13}$$

Här är v robotens hastighet, a accelerationen och δ styrvinkeln. Dessa begränsningar innebär att roboten endast kan ha icke-negativ hastighet, samt att acceleration

och styrvinkel hålls inom valda maximala gränser. Hastigheten begränsas till ickenegativa värden för att arbetet fokuseras på framåtriktad rörelse, vilket innebär att robotarna inte kan backa i modellen. För att tillåta backning hade hastighetsbegränsningen behövt ändras till både negativa och positiva hastigheter.

För att uppdatera robotens tillstånd används integrationsmetoden RK4, enligt beskrivning i avsnitt 2.2. Detta är särskilt viktigt vid snabba hastighetsförändringar eller när roboten svänger.

Om robotens kontinuerliga modell skrivs som:

$$\dot{x} = f(x, u) \tag{3.14}$$

där x är robotens tillstånd och u är styrsignalen, kan ett tidssteg med RK4 skrivas som:

$$k_1 = f(x_k, u_k) \tag{3.15}$$

$$k_2 = f\left(x_k + \frac{\Delta t}{2} k_1, u_k\right) \tag{3.16}$$

$$k_3 = f\left(x_k + \frac{\Delta t}{2} k_2, u_k\right) \tag{3.17}$$

$$k_4 = f\left(x_k + \Delta t\, k_3, u_k\right) \tag{3.18}$$

Robotens nästa tillstånd beräknas sedan enligt:

$$x_{k+1} = x_k + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{3.19}$$

Här är Δt tidsstegets längd. På detta sätt beräknas robotens nästa tillstånd genom flera uppskattningar av derivatan inom samma tidssteg.

3.4 Referensbana och referenspunkter

I simuleringen används en tidsberoende referensfunktion för varje robot. Referensfunktionen betecknas som $r(t)$ och returnerar det referenstillstånd som roboten ska följa vid tiden t:

$$r(t) = \Big[x_{\text{ref}}(t), \quad y_{\text{ref}}(t), \quad \psi_{\text{ref}}(t), \quad v_{\text{ref}}(t)\Big]^T$$

Här beskriver $x_{\text{ref}}(t)$ och $y_{\text{ref}}(t)$ robotens önskade position, $\psi_{\text{ref}}(t)$ den önskade riktningen och $v_{\text{ref}}(t)$ den önskade hastigheten vid tiden t.

Vid varje tidssteg hämtas ett aktuellt referenstillstånd från $r(t)$. Detta referenstillstånd används sedan av LQR och MPC-regulatorerna för att beräkna styrsignalerna. På detta sätt blir regleringen oberoende av hur referensbanan ursprungligen har skapats. Det viktiga för implementationen är att referensen kan beskrivas som en funktion av tiden.

För att beräkna referensriktningen används riktningen mellan två efterföljande referenspunkter:

$$\psi_{\text{ref}}(t) = \text{atan2}\left(y_{\text{ref}}(t + \Delta t) - y_{\text{ref}}(t), x_{\text{ref}}(t + \Delta t) - x_{\text{ref}}(t)\right)$$

Referenshastigheten kan beräknas från förändringen i referensposition mellan två tidpunkter:

$$v_{\text{ref}}(t) = \frac{\sqrt{(x_{\text{ref}}(t + \Delta t) - x_{\text{ref}}(t))^2 + (y_{\text{ref}}(t + \Delta t) - y_{\text{ref}}(t))^2}}{\Delta t}$$

Dessa värden används som indata till regulatorerna vid varje simuleringssteg.

4

Analys

I detta kapitel beskrivs experimenten, hur metoderna jämförs och vilka mått som används för att bedöma deras prestanda. De fyra reglerstrategierna analyseras utifrån referensföljning, antal kollisioner och hur säkerhetsavståndet påverkas när Control Barrier Functions används.

4.1 Experimenten

För att jämföra reglerstrategierna användes totalt 12 testfall. Varje testfall innehåller en karta över miljön samt en plan som beskriver robotarnas startpunkter, målpunkter och rörelse över tid. Testfallen kördes med fyra reglerstrategier: LQR, LQR+CBF, MPC, och MPC+CBF. Detta innebär att totalt 48 simuleringar genomfördes.

Beskrivning	Värde
Antal testfall	12
Antal reglerstrategier	4
Totalt antal simuleringar	48
Antal robotar per testfall	5, 10, 25 eller 50
Totalt antal robotar	270
Reglerstrategier	LQR, LQR+CBF, MPC, MPC+CBF

Tabell 4.1: Sammanfattning av experimentens omfattning

Testfallen skiljer sig främst åt genom antal robotar och kartans storlek. De enklare testfallen innehåller 5 robotar, medan mer komplexa testfall innehåller 10, 25 eller 50 robotar. När antalet robotar ökar blir scenariot svårare, eftersom fler robotar rör sig samtidigt i samma miljö. Detta ökar risken för konflikter, kollisioner och större avvikelser från referensbanorna.

Parameter	Värde
Robotlängd L	1.0
Robotbredd W	0.5
Tidssteg T_s	0.1 s
Maxhastighet $v_{\max}$	2.0
Maxacceleration $a_{\max}$	1.5
Max styrvinkel $\delta_{\max}$	1.0 rad
Kollisionsradie	1.0
Säkerhetszonens radie	0.8
MPC-horisont N_{horizon}	15
Lookahead-tid	0.3 s
CBF-parameter α	0.5

Tabell 4.2: Gemensamma simuleringsparametrar

Parametrarna i tabellen beskriver de gemensamma inställningar som används i simuleringarna. L och W beskriver robotens längd och bredd. T_s är tidssteget och anger hur ofta robotens tillstånd uppdateras i simuleringen. $v_{\max}$, $a_{\max}$ och $\delta_{\max}$ anger de största tillåtna värdena för hastighet, acceleration och styrvinkel. Dessa begränsningar används för att roboten inte kan köra, accelerera eller svänga hur mycket som helst.

Kollisionsradien används för att avgöra när två robotar räknas som kolliderande. Säkerhetszonens radie används i CBF-delen för att försöka hålla robotarna på ett säkert avstånd från varandra. N_{horizon} anger hur många framtida tidssteg MPC tar hänsyn till vid optimeringen. Lookahead-tiden används för att välja en referenspunkt en kort bit framför roboten, medan CBF-parametern α påverkar hur snabbt systemet får närma sig säkerhetsgränse.

4.2 Analys av testfall

För att jämföra reglerstrategierna används samma testfall för LQR, LQR + CBF, MPC, och MPC + CBF. Testfallen beskriver hur flera robotar ska röra sig från sina startpositioner till sina målpositioner. På detta sätt kan metoderna jämföras under samma förutsättningar.

I de analyserade testfallen skapas referensbanorna från grafbaserade planer. Grafen består av noder och kanter, där noderna motsvarar positioner i planet och kanterna beskriver möjliga förflyttningar mellan dessa positioner. Figur 4.1 visar ett testscenario.

När en robot rör sig mellan två noder antas referensen gå längs en rak linje med konstant hastighet. Om startnoden har position $p_0 = [x_0 \ y_0]^T$ och slutnoden har position $p_1 = [x_1 \ y_1]^T$ kan referenspositionen beskrivas med linjär interpolation:

$$p_{\text{ref}}(t) = p_0 + s(t)(p_1 - p_0) \tag{4.1}$$

där $s(t)$ är en tidsberoende parameter som varierar med t över intervallet $[0, 1]$. När $s(t) = 0$ befinner sig roboten vid startnoden, och när $s(t) = 1$ har roboten

nått slutnoden. Värden mellan 0 och 1 beskriver robotens position mellan dessa två noder.

Referensriktningen beräknas från riktningen mellan start- och slutnoden:

$$\psi_{\text{ref}} = \arctan2\left(y_1 - y_0, x_1 - x_0\right) \tag{4.2}$$

Referenshastigheten bestäms av avståndet mellan noderna och tiden som finns tillgänglig för rörelsen:

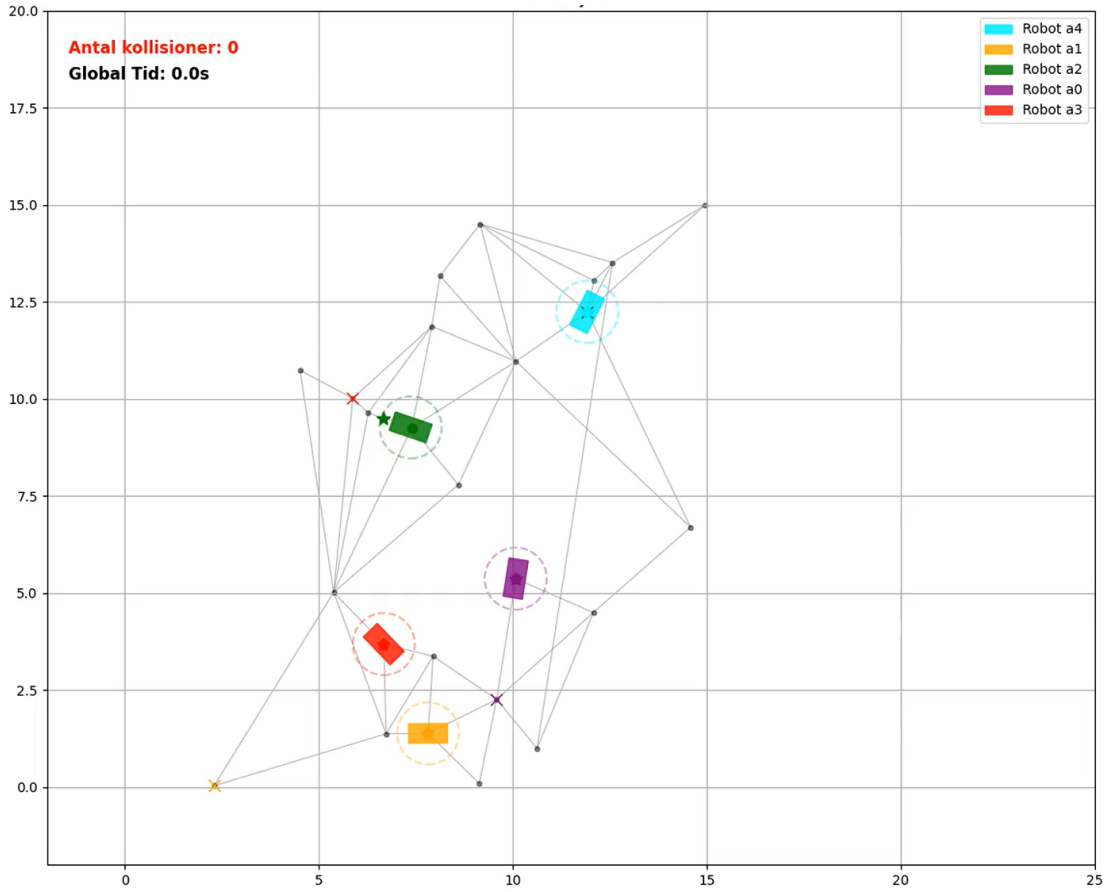
$$v_{\text{ref}} = \frac{\|p_1 - p_0\|}{t_1 - t_0} \tag{4.3}$$

där t_0 och t_1 är start- respektive sluttid för rörelsen. När roboten väntar vid en nod hålls referenspositionen konstant och referenshastigheten sätts till:

$$v_{\text{ref}} = 0 \tag{4.4}$$

Under väntan finns ingen ny rörelseriktning att beräkna, eftersom positionen är konstant. Därför används riktningen från den senaste rörelsen in till noden. Det innebär att roboten behåller samma referensriktning under väntan som den hade när den kom fram till noden. Detta är naturligt för den kinematiska cykelmodellen, eftersom roboten behöver röra sig för att kunna svänga.

För regulatorerna är det viktigaste att referensen kan beskrivas som en tidsberoende funktion $r(t)$. Funktionen ger robotens referensposition, referensriktning och referenshastighet vid varje tidpunkt. I testfallen skapas referensbanan $r(t)$ från grafbaserade JSON-filer. LQR och MPC använder denna referens för att bestämma hur roboten ska styras vid varje tidssteg. CBF används sedan som ett extra säkerhetslager för att minska risken för kollisioner mellan robotarna.



Figur 4.1: Figuren visar kartans noder och kanter, robotarnas positioner samt deras start- och målpositioner.

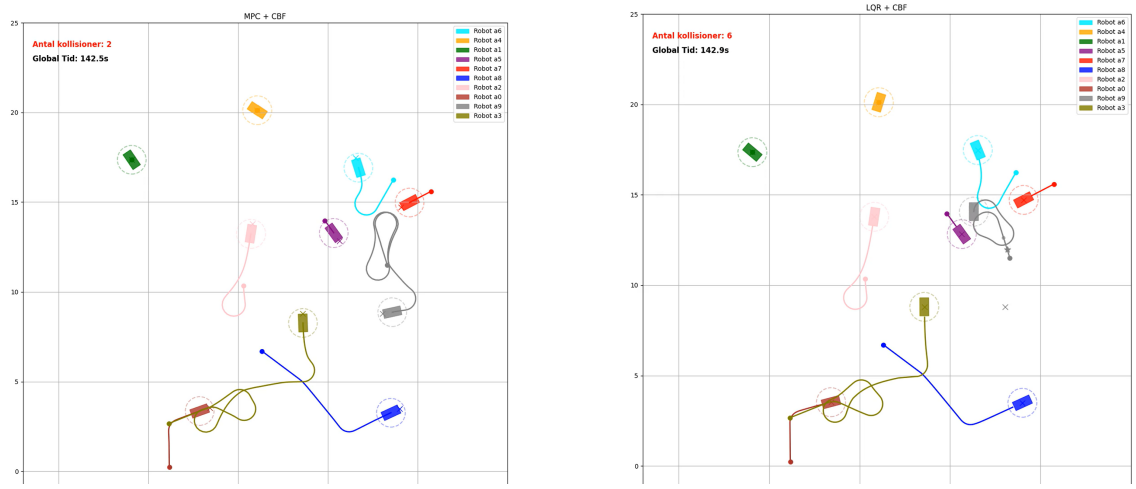
4.3 Datainsamling och verifiering

Under simuleringarna sparas data för varje robot för att kunna jämföra reglerstrategierna. Den sparade datan består av robotens position, riktning, hastighet och styrsignaler. För LQR sparas även sidofel och riktningsfel, medan MPC-data innehåller tillstånd och styrsignaler över tid.

Metoderna jämförs utifrån referensföljning, antal kollisioner och hur säkerhetsavståndet påverkas när CBF används. En kollision registreras när två robotar kommer närmare varandra än den valda kollisionsradien. Figur 4.2 visar hur robotarna rör sig i ett tätare scenario vid samma tidpunkt, där kollisioner kan uppstå. Det viktiga i figuren är att se hur robotarnas banor och kollisionsantal skiljer sig mellan metoderna. I MPC+CBF är antalet kollisioner lägre och flera robotar håller större avstånd till varandra. I LQR+CBF uppstår fler kollisioner. Figuren visar därför att säkerheten inte bara beror på CBF, utan också på hur väl grundregulatorn styr robotarna.

För CBF analyseras även barriärfunktionen $h(x)$. Om $h(x)$ är positiv befinner sig roboten utanför det osäkra området. Om $h(x)$ blir negativ betyder det att säkerhetsavståndet har brutits.

För att beskriva referensföljningen används måtten J_{robot}, J_{test} och J_{method}. Måttet J_{robot} beskriver medelpositionsfelet för en robot, J_{test} beskriver medelfelet för alla robotar i ett testfall och J_{method} beskriver medelfelet över 12 testfall. Tillsammans med antalet kollisioner används dessa mått för att bedöma vilken metod som fungerar bäst.



(a) MPC+CBF **(b)** LQR+CBF

Figur 4.2: Jämförelse vid samma tidpunkt i ett tätare multirobotscenario

För att jämföra hur bra Robot a2 följer sin referens används medelpositionsfelet. Först beräknas positionsfelet vid varje tidssteg enligt

$$e_{pos}(k) = \sqrt{(x(k) - x_{ref}(k))^2 + (y(k) - y_{ref}(k))^2}$$

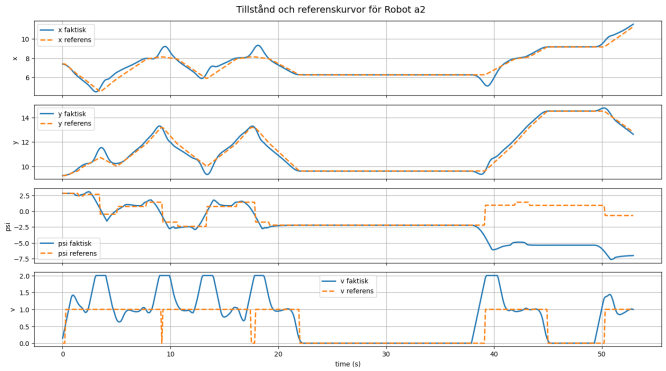
Därefter beräknas medelvärdet över alla tidssteg:

$$J_{robot} = \frac{1}{N} \sum_{k=1}^{N} e_{pos}(k)$$

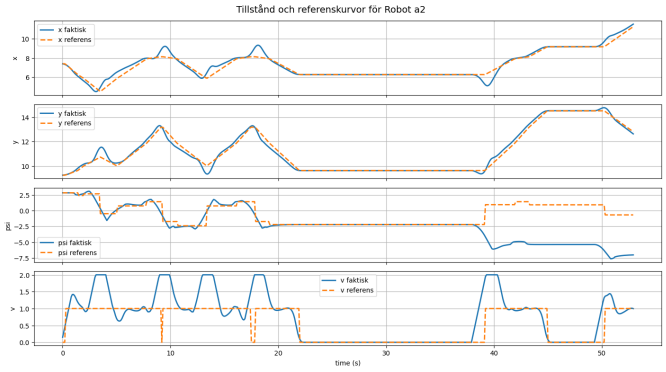
Detta värde visar hur långt roboten i genomsnitt ligger från sin referensbana. Ett lägre värde betyder bättre referensföljning.

Figur 4.3 visar ett exempel på referensföljningen för Robot a2 med de fyra reglerstrategierna. I varje delfigur jämförs robotens faktiska tillstånd med referenstillståndet över tid. De blå kurvorna visar robotens faktiska värden, medan de orange streckade kurvorna visar referensen.

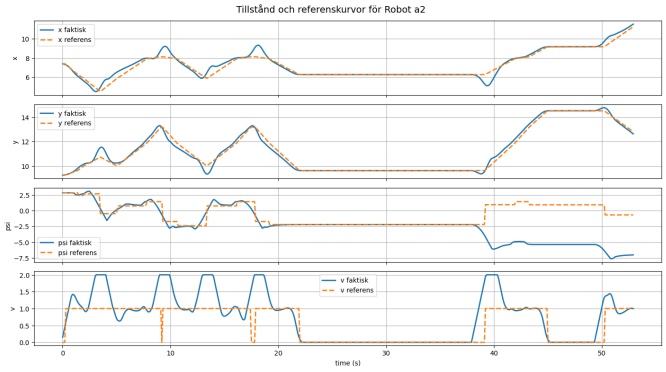
Det viktiga att observera är hur de blå kurvorna ligger nära de orange kurvorna. När kurvorna ligger nära varandra följer roboten referensen väl, medan större skillnader mellan kurvorna innebär större avvikelse. Figuren används därför som ett visuellt exempel på det positionsfel.



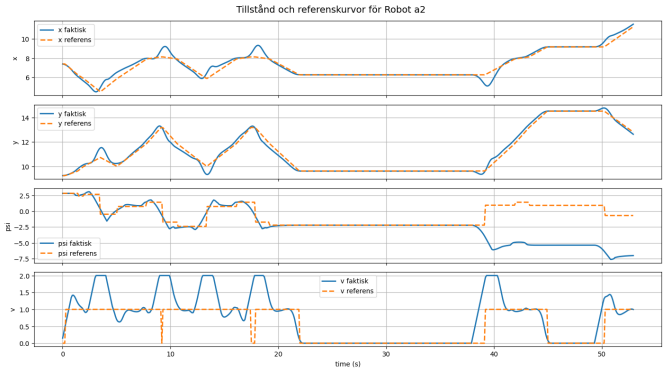
(a) LQR+CBF



(b) LQR



(c) MPC



(d) MPC+CBF

Figur 4.3: Jämförelse mellan faktisk rörelse och referenskurvor för Robot a2 med de fyra reglerstrategierna

För att bedöma hur bra alla robotar följer sina referenser i ett test beräknas först J_{robot} för varje robot. Därefter beräknas medelvärdet över alla robotar:

$$J_{test} = \frac{1}{M} \sum_{i=1}^{M} J_{robot,i}$$

Måttet J_{method} används för att beskriva metodens referensfel över 12 testfall. Först beräknas J_{test} för varje testfall, vilket är medelpositionsfelet över alla robotar i testet. Därefter beräknas medelvärdet av J_{test} över 12 testfall:

$$J_{method} = \frac{1}{T} \sum_{j=1}^{T} J_{test,j}$$

där T är antalet testfall.

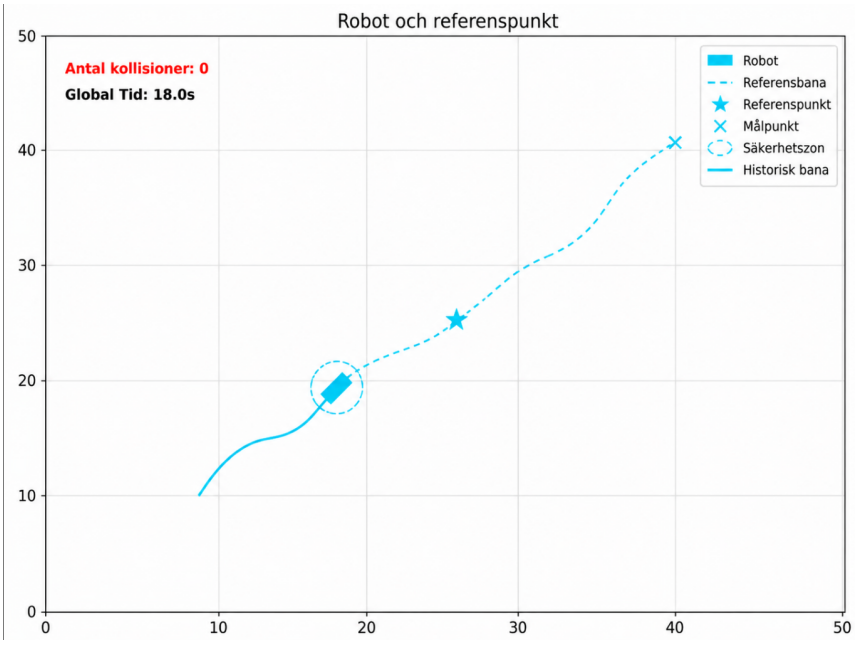
5

Resultat

I detta kapitel presenteras resultaten från simuleringarna med fokus på referensföljning och säkerhet. Resultaten baseras på de 12 testfall som beskrevs i kapitel 4, där varje testfall kördes med fyra reglerstrategier

5.1 Prestanda och banföljning

Figur 5.1 visar ett exempel på hur referensföljningen visualiseras vid en viss tidpunkt i simuleringen. Referenspunkten ligger framför roboten och används för att styra roboten längs den tidsberoende referensbanan. Den streckade linjen visar den planerade referensbanan, medan den heldragna linjen visar robotens faktiska rörelse fram till den aktuella tidpunkten.



Figur 5.1: Exempel på hur roboten följer en referensbana med en referenspunkt framför sig

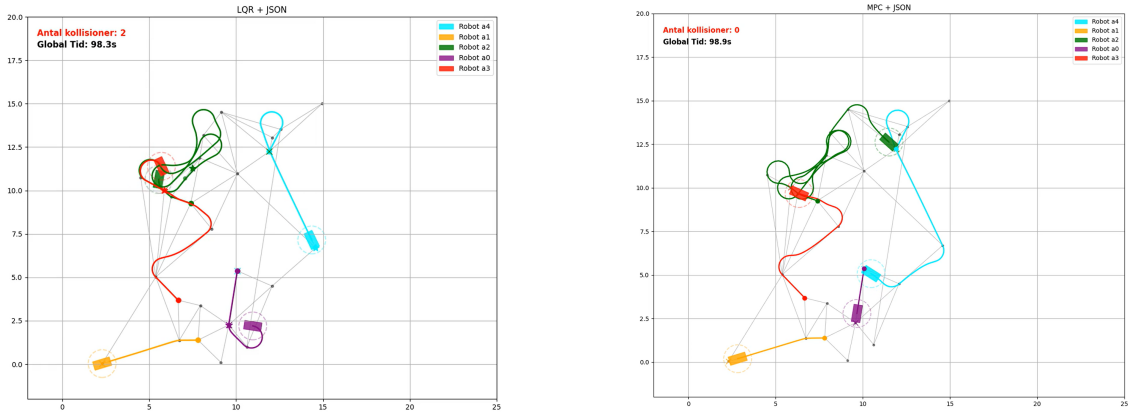
Figur 5.2 visar ögonblicksbilder från samma scenario för LQR och MPC vid samma tidpunkt. Bilderna används som ett visuellt exempel på skillnaden mellan metoderna. I LQR syns mer slingrande rörelser och större avvikelser från den planerade

referensstrukturen, medan MPC visar ett mer kontrollerat beteende. Figurerna används endast som illustrationer, och det dynamiska förloppet visas tydligare i animationsvideorna som finns på GitHub[1]. Den huvudsakliga jämförelsen baseras på de numeriska måtten. Tabell 5.1 sammanfattar resultaten för de fyra reglerstrategierna. Tabellen visar både referensföljning och totalt antal kollisioner för varje metod.

Metod	J_{robot}	J_{test}	J_{method}	**Totalt antal kollisioner**
LQR	3.534	1.872	0.896	75
LQR med CBF	3.518	1.444	0.728	55
MPC	0.287	0.106	0.088	44
MPC med CBF	0.287	0.109	0.076	30

Tabell 5.1: Sammanfattning av referensföljning och kollisioner för de fyra reglerstrategierna

Resultaten visar en tydlig skillnad mellan LQR och MPC metoder. LQR hade svårast att följa referensbanorna och fick det högsta genomsnittliga felet, $J_{\text{method}} = 0.896$. När CBF lades till LQR minskade felet till $J_{\text{method}} = 0.728$, vilket visar en viss förbättring, men metoden hade fortfarande problem i mer komplexa scenarier. MPC gav betydligt bättre referensföljning än LQR. För MPC blev $J_{\text{method}} = 0.088$, vilket visar att metoden kunde följa referensbanorna med mycket mindre avvikelse. Detta beror på att MPC optimerar styrsignalerna över en framtida tidshorisont och därmed kan planera rörelsen mer effektivt.



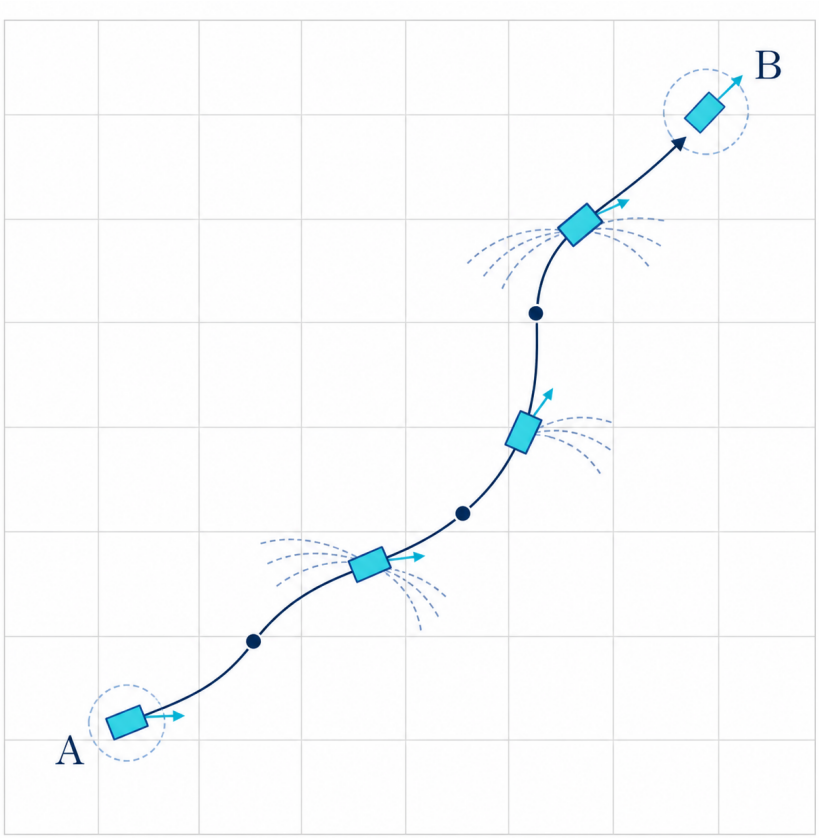
Figur 5.2: Ögonblicksbilder från samma scenario för MPC och LQR vid samma tidpunkt

Figur 5.3 illustrerar schematiskt hur roboten följer en kurvad bana från startpunkt A till målpunkt B.

Det bästa värdet erhölls för MPC + CBF, där $J_{\text{method}} = 0.076$. Detta innebär att kombinationen av MPC och CBF gav den bästa genomsnittliga referensföljningen

[1] `https://github.com/Yaman30/MAPF`

över testfallen. Skillnaden i referensföljning mellan MPC och MPC + CBF var relativt liten. Det visar att CBF inte främst förbättrar banföljningen, utan framför allt bidrar till ökad säkerhet genom att minska risken för kollisioner.



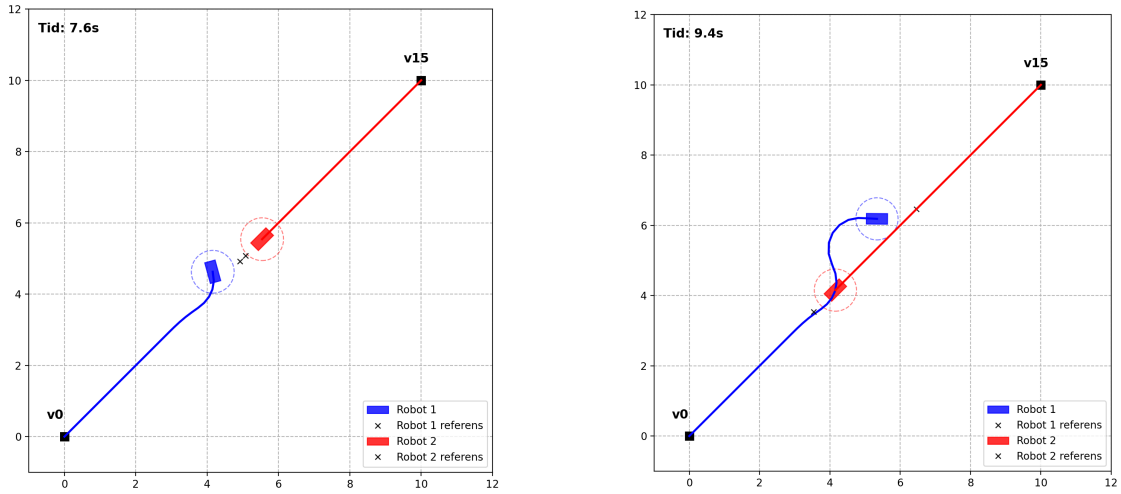
Figur 5.3: Robotens rörelse genom referensbanan från A till B

5.2 Säkerhet och kollisioner

Antalet kollisioner minskade när CBF användes. Resultaten bör tolkas med hänsyn till att varje metod testades på totalt 270 robotkörningar. LQR gav totalt 75 kollisioner, medan LQR + CBF minskade detta till 55 kollisioner. På samma sätt minskade antalet kollisioner från 44 för MPC till 30 för MPC + CBF. Figur 5.4 visar ett exempel där två robotar styrs med CBF och håller avstånd till varandra med hjälp av säkerhetszoner.

Detta visar att CBF förbättrade säkerheten, även om metoden inte helt eliminerade alla kollisioner i de genomförda simuleringarna. Resultaten visar därför att CBF fungerar som ett säkerhetslager som kan minska risken för kollisioner. Samtidigt beror säkerheten även på hur den valda regulatorn styr roboten från början och på testfallets komplexitet, till exempel hur många robotar som rör sig samtidigt och hur ofta de kommer nära varandra.

MPC + CBF gav bäst säkerhetsresultat, eftersom metoden hade lägst antal kollisioner.



Figur 5.4: Två robotar undviker kollisionen

5.3 Sammanfattning av resultat

Sammanfattningsvis visar resultaten att LQR är enkel och kan fungera i enklare fall, men att metoden får problem när banorna innehåller skarpa svängar eller när flera robotar rör sig nära varandra. MPC gav betydligt bättre referensföljning och mer stabila rörelser.

LQR hade störst svårigheter att följa referensbanorna och gav flest kollisioner. När CBF lades till LQR minskade antalet kollisioner och referensföljningen förbättrades något, men metoden hade fortfarande problem i svårare testfall. MPC gav mycket bättre referensföljning och mjukare rörelser än LQR. MPC + CBF gav det bästa helhetsresultatet, eftersom den hade lägst J_{method} värde och lägst antal kollisioner. Det visar att CBF förbättrade säkerheten utan att försämra banföljningen i de testade scenarierna.

6

Diskussion och slutsats

I detta kapitel diskuteras de erhållna resultaten i relation till projektets syfte och frågeställningar. Vidare dras slutsatser kring metodernas lämplighet samt förslag ges på framtida arbete.

6.1 Diskussion av resultat

Resultaten visar att valet av reglerstrategi har stor påverkan på både referensföljning och säkerhet. LQR är enkel att implementera och kräver relativt låg beräkningskraft, men metoden får tydliga problem i mer komplexa scenarier. Eftersom LQR är en reaktiv metod kan den inte planera framåt på samma sätt som MPC. Detta leder till större avvikelser från referensbanan i vissa scenarier, särskilt vid skarpa svängar eller när robotarna behöver göra snabba riktningsändringar. Ett exempel visas i Figur 5.2, där LQR ger mer slingrande rörelser än MPC. Figuren visas tydligare i animationsvideorna på GitHub[1]. Skillnaden får stöd också av de numeriska resultaten i Tabell 5.1, där LQR har högre J_{method} än MPC.

MPC gav betydligt bättre resultat när det gäller referensföljning. Det låga värdet på J_{method} visar att MPC-baserade metoder kunde följa referensbanorna mer noggrant än LQR-baserade metoder. Detta beror på att MPC tar hänsyn till både systemets modell och begränsningar över en framtida tidshorisont. Resultatet stämmer även med tidigare studier där MPC lyfts fram som en lämplig metod för system med kinematiska begränsningar och styrbegränsningar.

Införandet av CBF förbättrade säkerheten genom att minska antalet kollisioner. För LQR minskade antalet kollisioner från 75 till 55, och för MPC minskade antalet kollisioner från 44 till 30. Detta visar att CBF fungerar som ett säkerhetslager som kan minska risken för kollisioner.

Samtidigt visar resultaten att CBF inte helt eliminerade alla kollisioner i simuleringarna. Detta innebär att CBF i denna implementation inte kan ses som en fullständig garanti för kollisionsfrihet, utan snarare som en metod som förbättrar säkerheten. En förklaring är att säkerhetsfiltret endast justerar styrsignalen lokalt och inte gör en fullständig omplanering av robotarnas banor. Vid hög agenttäthet kan detta leda till situationer där robotarna hamnar för nära varandra innan styrsignalen hinner korrigeras tillräckligt.

[1] `https://github.com/Yaman30/MAPF`

En annan möjlig förklaring till att CBF inte eliminerade alla kollisioner är att flera CBF-villkor kan vara svåra att uppfylla samtidigt. Detta är ett känt problem i litteraturen. Även om de enskilda CBF-villkoren kan uppfyllas var för sig, kan kombinationen av villkoren göra att det saknas en styrsignal som uppfyller alla krav samtidigt. Problemet blir särskilt tydligt när roboten har begränsningar på acceleration och styrvinkel.

Breeden och Panagou [13] visar att två CBF-villkor kan ha varsin möjlig mängd styrsignaler, men att dessa mängder ändå kan sakna gemensamt snitt när villkoren kombineras. Det innebär att det kvadratiska optimeringsproblemet inte får någon lösning. Zhao et al. [11]beskriver också kombinationen av flera CBF-villkor som en möjlig orsak till att CBF-QP-problem kan bli olösbara, särskilt tillsammans med aktuatorbegränsningar och olämpligt valda parametrar.

Detta kan förklara varför CBF i detta arbete minskade antalet kollisioner men inte eliminerade dem helt. En mer detaljerad analys av hur flera CBF-villkor kan kombineras på ett lösbart sätt lämnas därför som framtida arbete.

Resultaten visar också att en diskret MAPF-plan inte automatiskt är direkt genomförbar för robotar med kinematiska begränsningar. De planerade banorna måste översättas till rörelser som robotmodellen faktiskt kan följa. Detta motiverar användningen av regulatorer som LQR och MPC, där särskilt MPC visade sig vara bättre lämpad för att hantera skarpa svängar, begränsningar och komplexa multi-robot-scenarier.

Den bästa helhetslösningen i detta arbete var MPC + CBF. Tabell 5.1 visar att metoden gav lägst genomsnittligt referensfel och lägst antal kollisioner. Jämfört med vanlig MPC blev referensfelet något lägre när CBF lades till, men skillnaden var liten. Den tydligaste förbättringen var att antalet kollisioner minskade. Därför kan CBF främst ses som ett säkerhetslager utan att försämra robotarnas förmåga att följa referensbanan.

6.2 Framtida arbete

Trots de goda resultaten finns det flera möjliga riktningar för vidareutveckling av systemet.

En naturlig fortsättning är att implementera och testa systemet på fysisk hårdvara. I detta arbete har alla experiment genomförts i simulering, vilket innebär att effekter som sensormätfel, fördröjningar och modellfel inte har beaktats. En implementation på riktiga robotar skulle därför ge en mer realistisk bild av systemets prestanda.

En annan möjlig utveckling är att förbättra hanteringen av lokala låsningar (deadlocks) som kan uppstå vid användning av CBF i täta miljöer. Detta skulle kunna göras genom att kombinera CBF med mer avancerade samordningsstrategier mellan robotar, där besluten inte enbart baseras på lokala säkerhetsvillkor.

En ytterligare vidareutveckling är att förbättra själva CBF formuleringen så att den kan hantera flera robotar mer tillförlitligt i täta scenarier. I detta arbete används CBF som ett lokalt säkerhetsfilter, men framtida arbete kan undersöka tidsberoende CBF-villkor och mer strukturerade metoder för att kombinera flera CBF-villkor sam-

tidigt. Detta skulle kunna förbättra systemets förmåga att förutse krockrisker och därmed minska risken för att robotarna hamnar i situationer där säkerhetsvillkoren inte kan uppfyllas.

Slutligen kan mer avancerade modeller undersökas, där dynamiska effekter såsom massa, tröghet och friktion inkluderas. Detta skulle ge en mer realistisk representation av robotens rörelse och möjliggöra en mer exakt analys av regulatorernas prestanda.

6.3 Slutsats

Syftet med detta arbete var att designa och utvärdera reglersystem för banföljning av flera robotar med fokus på precision och säkerhet. De undersökta metoderna var LQR, MPC, LQR + CBF, och MPC + CBF.

Resultaten visar att LQR fungerar i enklare scenarier, men att metoden har tydliga begränsningar när banorna blir mer komplexa eller när flera robotar rör sig nära varandra. LQR gav högst genomsnittligt referensfel och flest kollisioner.

MPC gav betydligt bättre banföljning än LQR. Genom att optimera över en framtida tidshorisont kunde MPC hantera referensbanan och systemets begränsningar på ett mer effektivt sätt. Detta gav lägre spårningsfel och mer stabila rörelser.

CBF förbättrade säkerheten genom att minska antalet kollisioner för både LQR och MPC. Däremot eliminerades inte alla kollisioner i simuleringarna, vilket visar att CBF i denna implementation fungerar som ett förbättrande säkerhetslager snarare än en fullständig garanti för kollisionsfrihet.

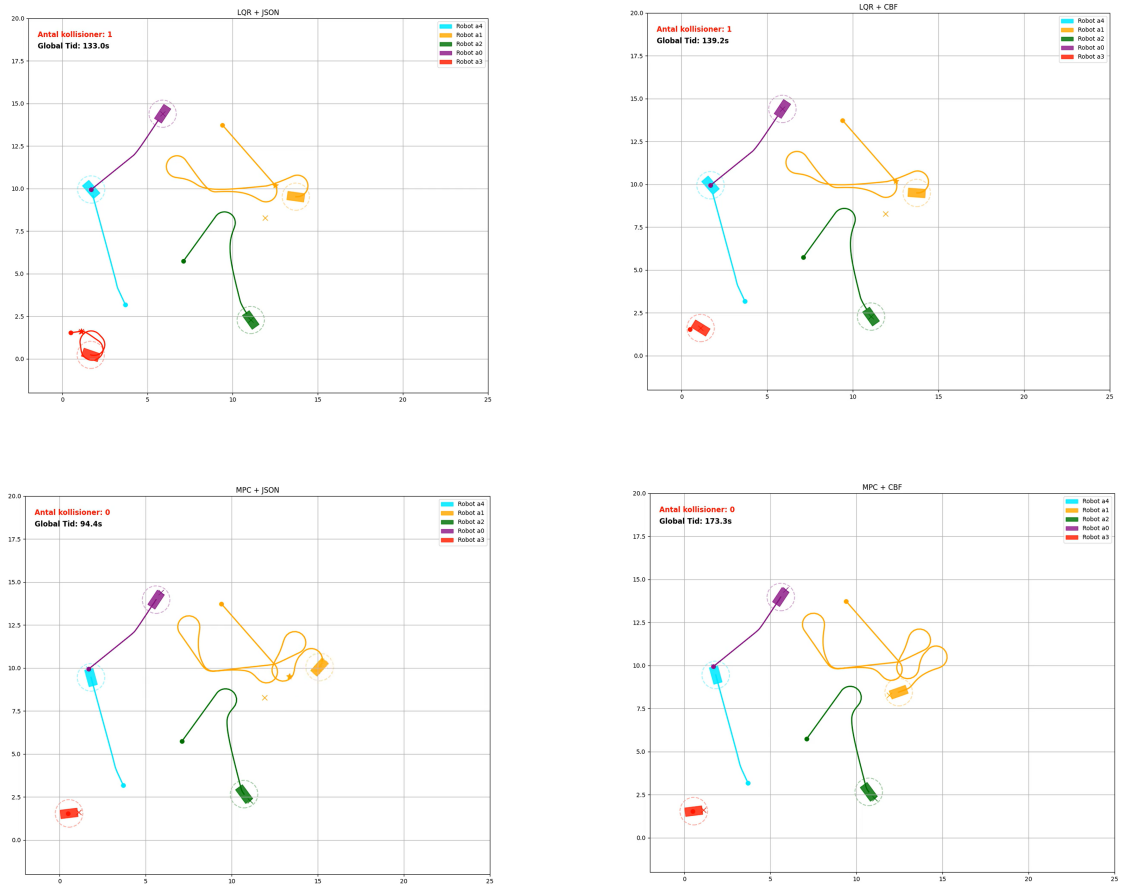
Den bästa helhetsmetoden var MPC + CBF. Denna metod gav lägst genomsnittligt referensfel och lägst antal kollisioner. Slutsatsen är därför att MPC + CBF är den mest lämpliga metoden i detta arbete för att kombinera noggrann banföljning med förbättrad säkerhet i multirobotsimulering.

7

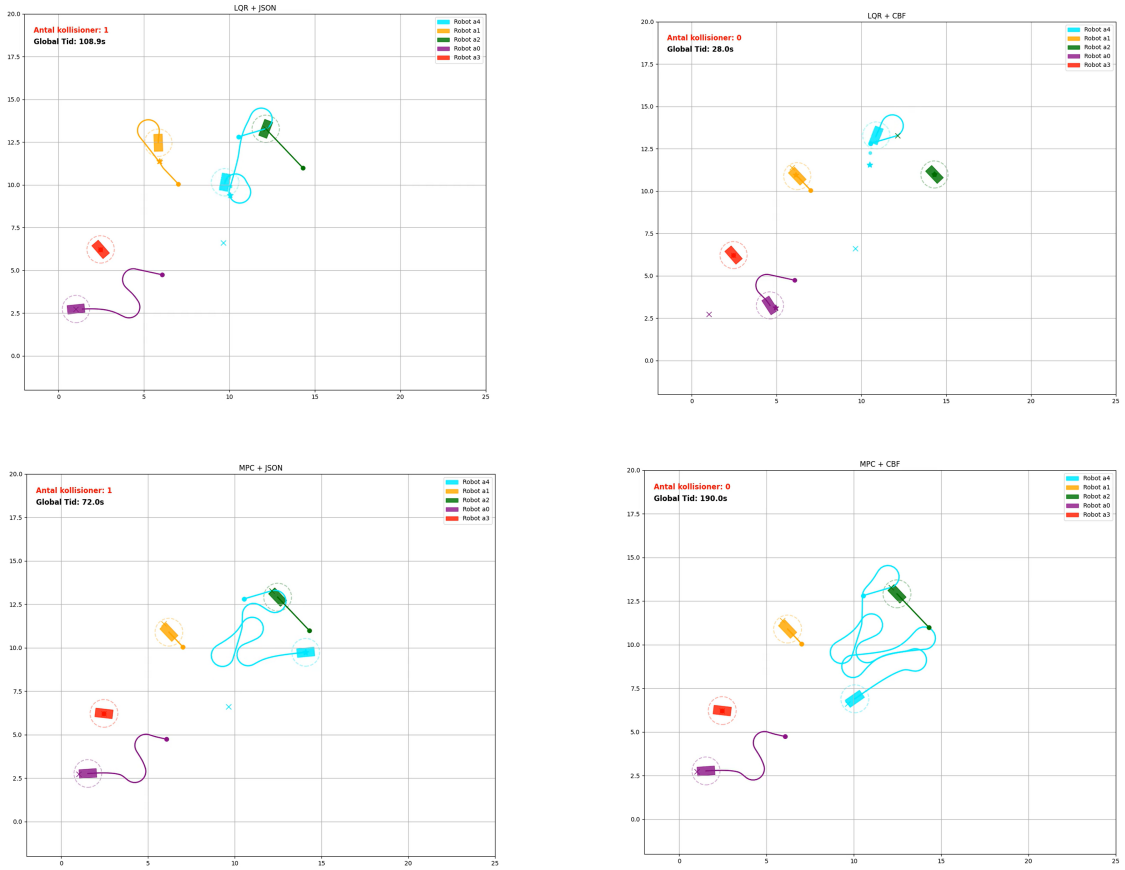
Bilagor

I denna bilaga presenteras kompletterande grafer och figurer för de olika simuleringsscenariern. Här finns även en länk till GitHub-sida, där projektets kod och en README-fil finns tillgängliga.

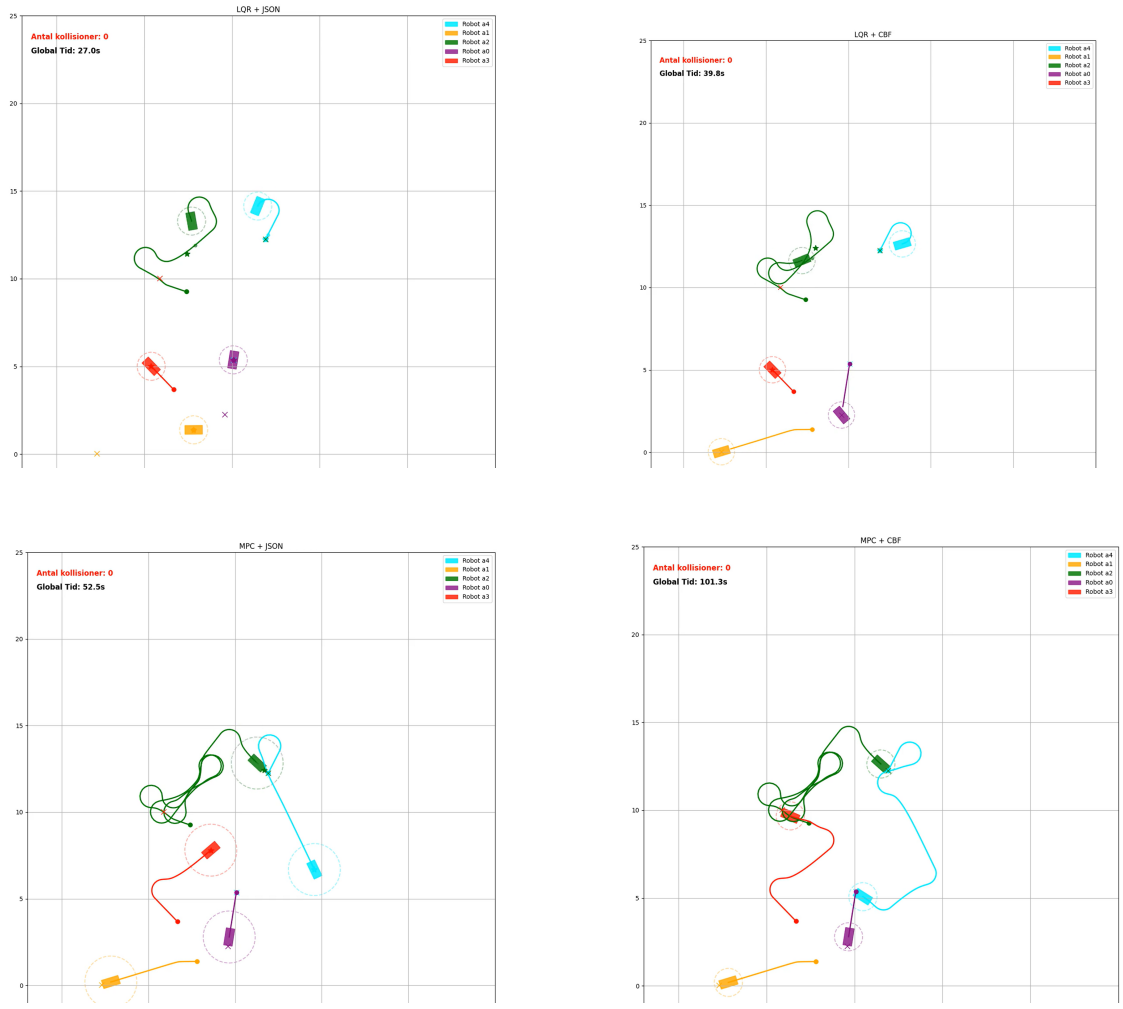
7.1 Figurer från simuleringsscenarier



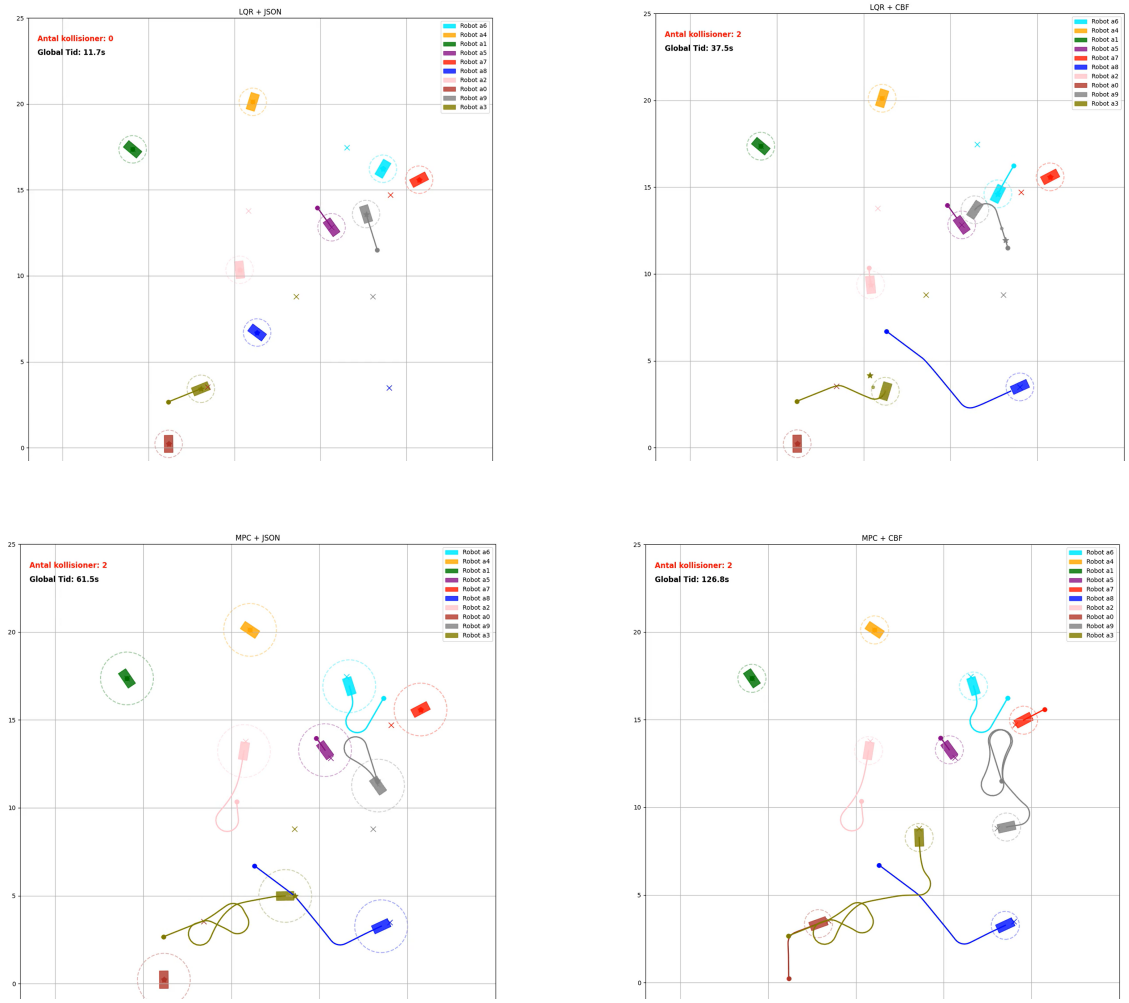
Figur 7.1


LQR + JSON
Antal kollisioner: 1
Global Tid: 108.9s
LQR + CBF
Antal kollisioner: 0
Global Tid: 28.0s
MPC + JSON
Antal kollisioner: 1
Global Tid: 72.0s
MPC + CBF
Antal kollisioner: 0
Global Tid: 190.0s
Robot a4
Robot a1
Robot a2
Robot a0
Robot a3

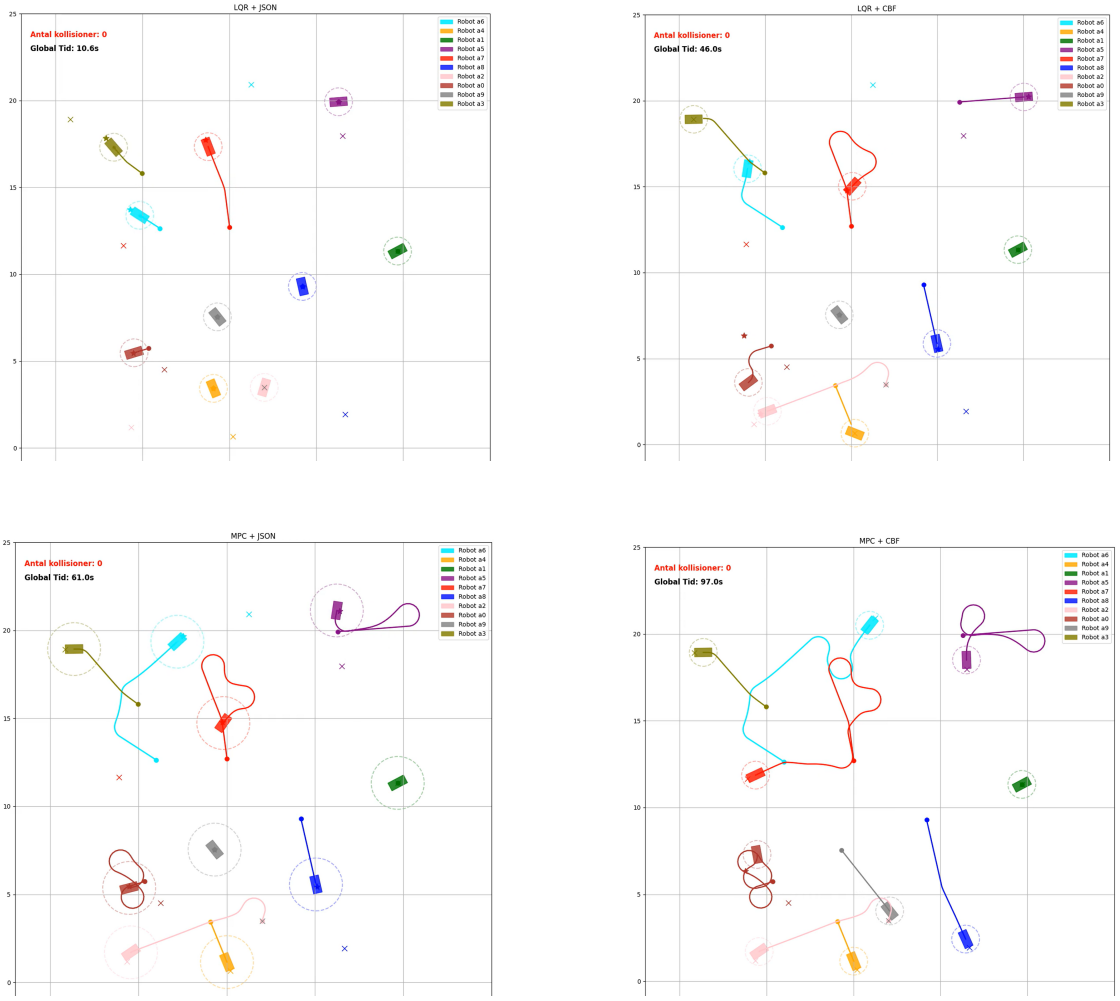
Figur 7.2


LQR + JSON
Antal kollisioner: 0
Global Tid: 27.0s
LQR + CBF
Antal kollisioner: 0
Global Tid: 39.8s
MPC + JSON
Antal kollisioner: 0
Global Tid: 52.5s
MPC + CBF
Antal kollisioner: 0
Global Tid: 101.3s
Robot a4
Robot a1
Robot a2
Robot a0
Robot a3

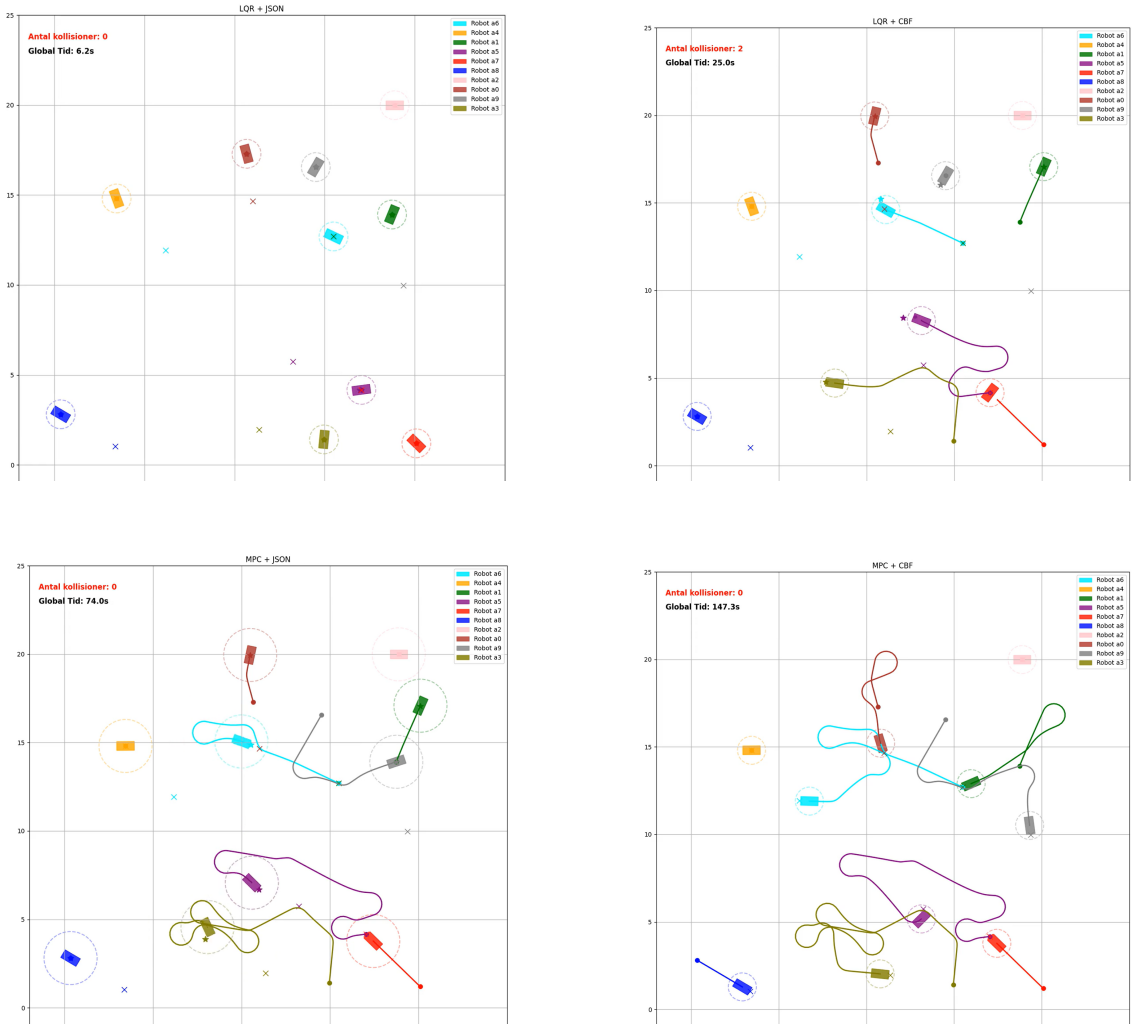
Figur 7.3


LQR + JSON
Antal kollisioner: 0
Global Tid: 11.7s
LQR + CBF
Antal kollisioner: 2
Global Tid: 37.5s
MPC + JSON
Antal kollisioner: 2
Global Tid: 61.5s
MPC + CBF
Antal kollisioner: 2
Global Tid: 126.8s
Robot a6
Robot a4
Robot a1
Robot a5
Robot a7
Robot a8
Robot a2
Robot a0
Robot a9
Robot a3

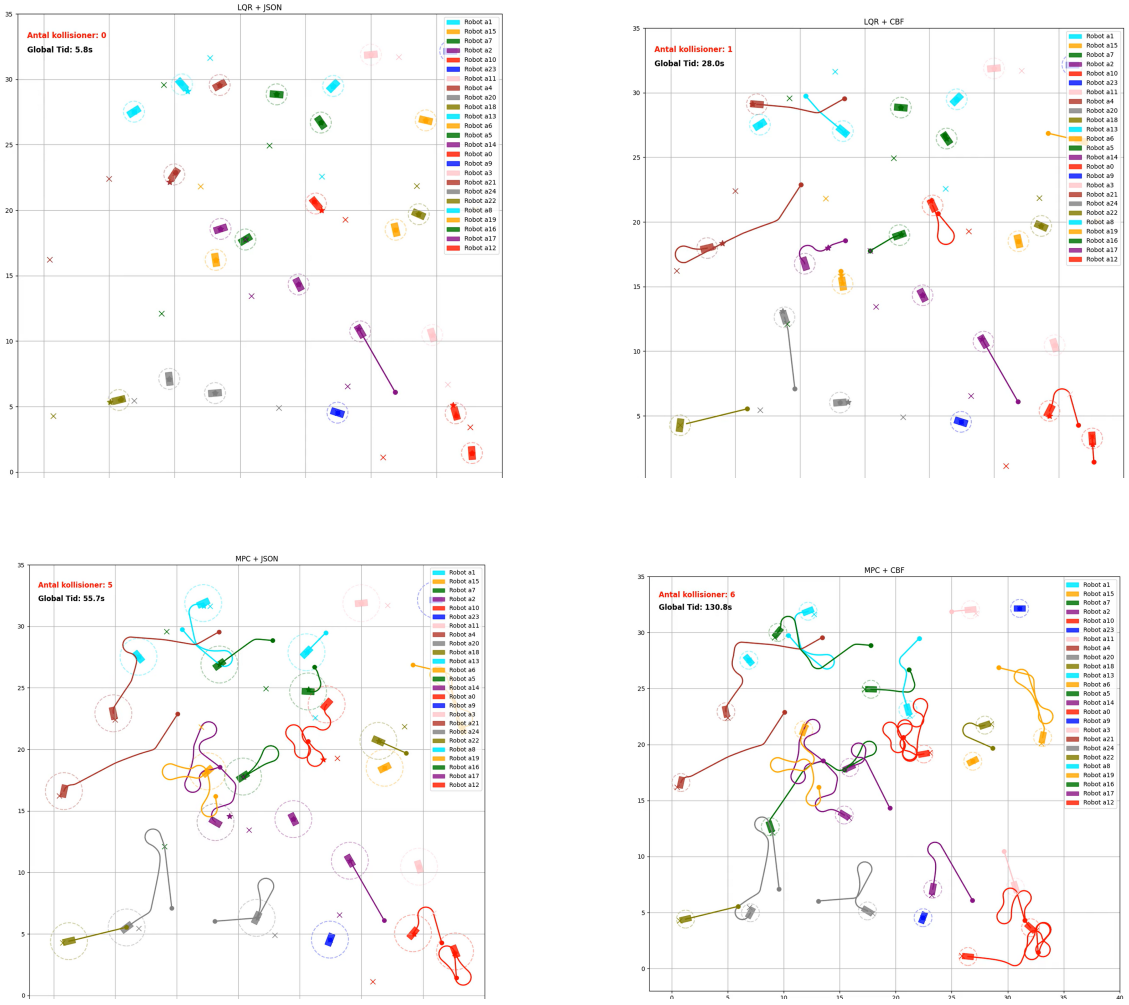
Figur 7.4


LQR + JSON
Antal kollisioner: 0
Global Tid: 10.6s
LQR + CBF
Antal kollisioner: 0
Global Tid: 46.0s
MPC + JSON
Antal kollisioner: 0
Global Tid: 61.0s
MPC + CBF
Antal kollisioner: 0
Global Tid: 97.0s
Robot a6
Robot a4
Robot a1
Robot a5
Robot a7
Robot a8
Robot a2
Robot a0
Robot a9
Robot a3

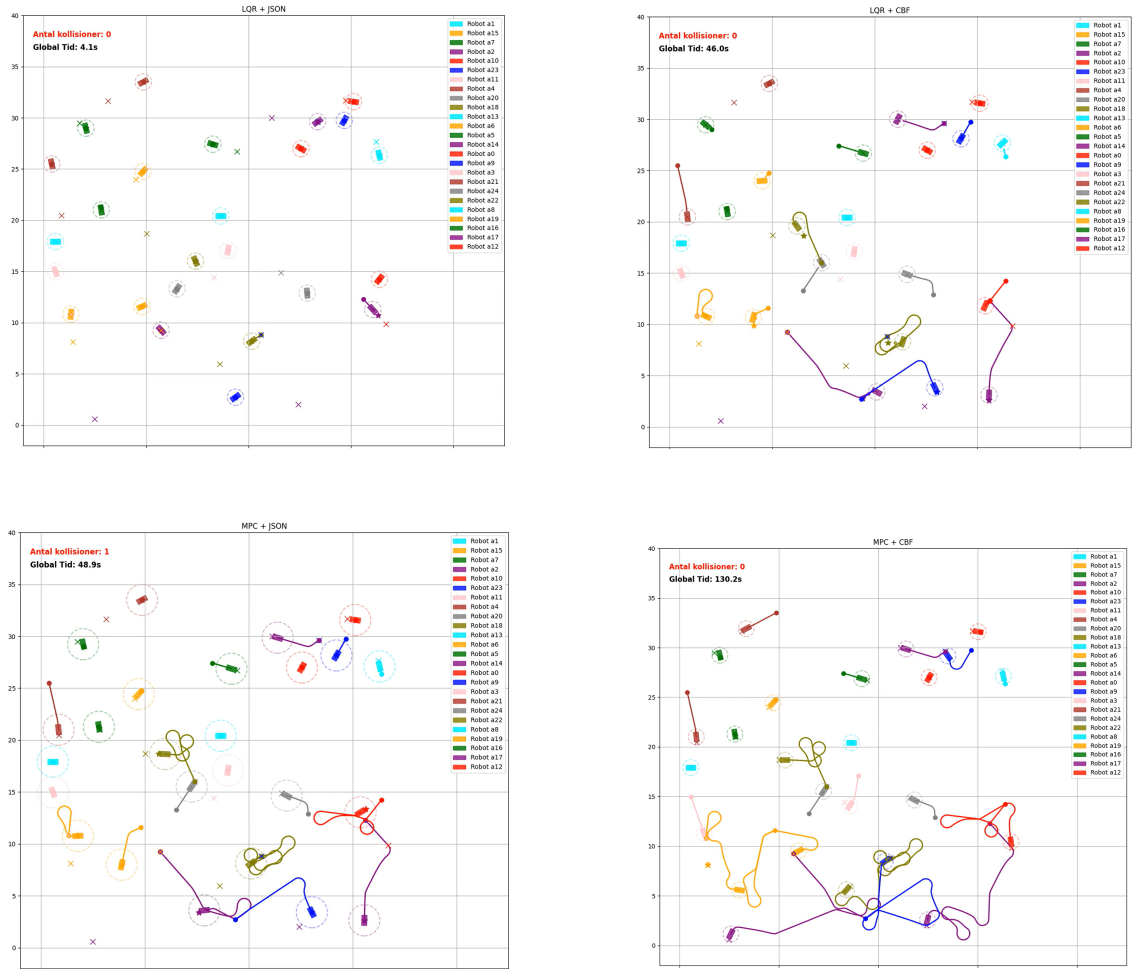
Figur 7.5


LQR + JSON
Antal kollisioner: 0
Global Tid: 6.2s
LQR + CBF
Antal kollisioner: 2
Global Tid: 25.0s
MPC + JSON
Antal kollisioner: 0
Global Tid: 74.0s
MPC + CBF
Antal kollisioner: 0
Global Tid: 147.3s
Robot a6
Robot a4
Robot a1
Robot a5
Robot a7
Robot a8
Robot a2
Robot a0
Robot a9
Robot a3

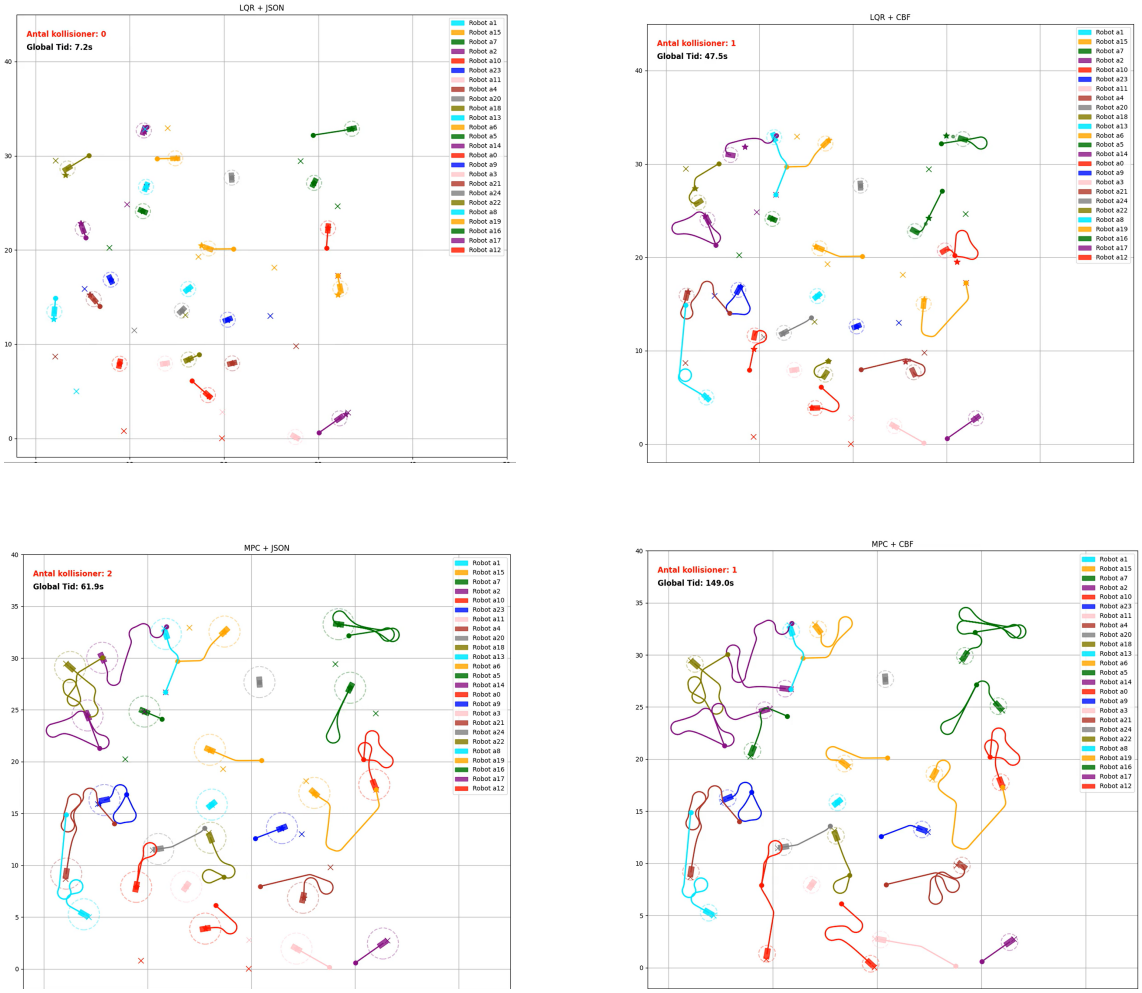
Figur 7.6


LQR + JSON
Antal kollisioner: 0
Global Tid: 5.8s
LQR + CBF
Antal kollisioner: 1
Global Tid: 28.0s
MPC + JSON
Antal kollisioner: 5
Global Tid: 55.7s
MPC + CBF
Antal kollisioner: 6
Global Tid: 130.8s

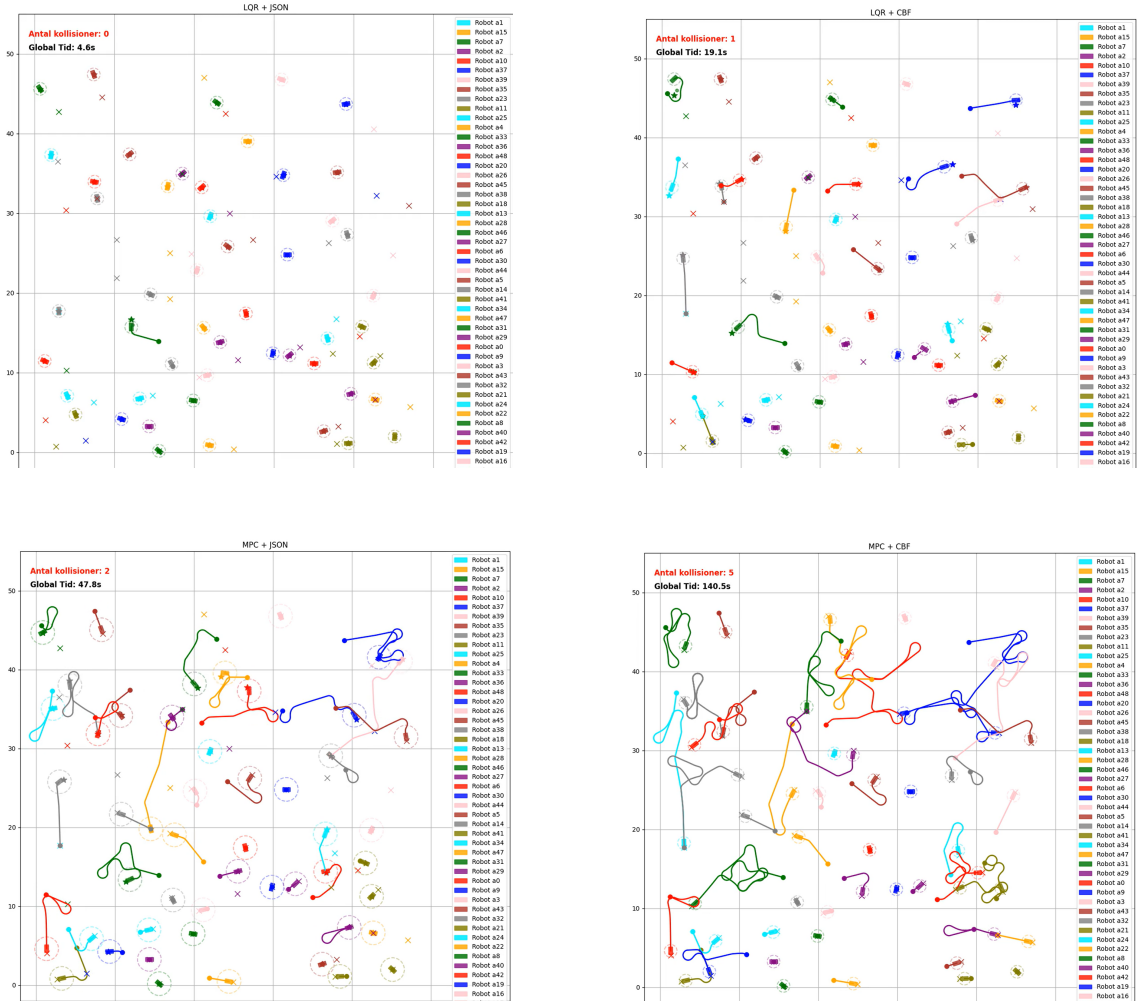
Figur 7.7


LQR + JSON
Antal kollisioner: 0
Global Tid: 4.1s
LQR + CBF
Antal kollisioner: 0
Global Tid: 46.0s
MPC + JSON
Antal kollisioner: 1
Global Tid: 48.9s
MPC + CBF
Antal kollisioner: 0
Global Tid: 130.2s

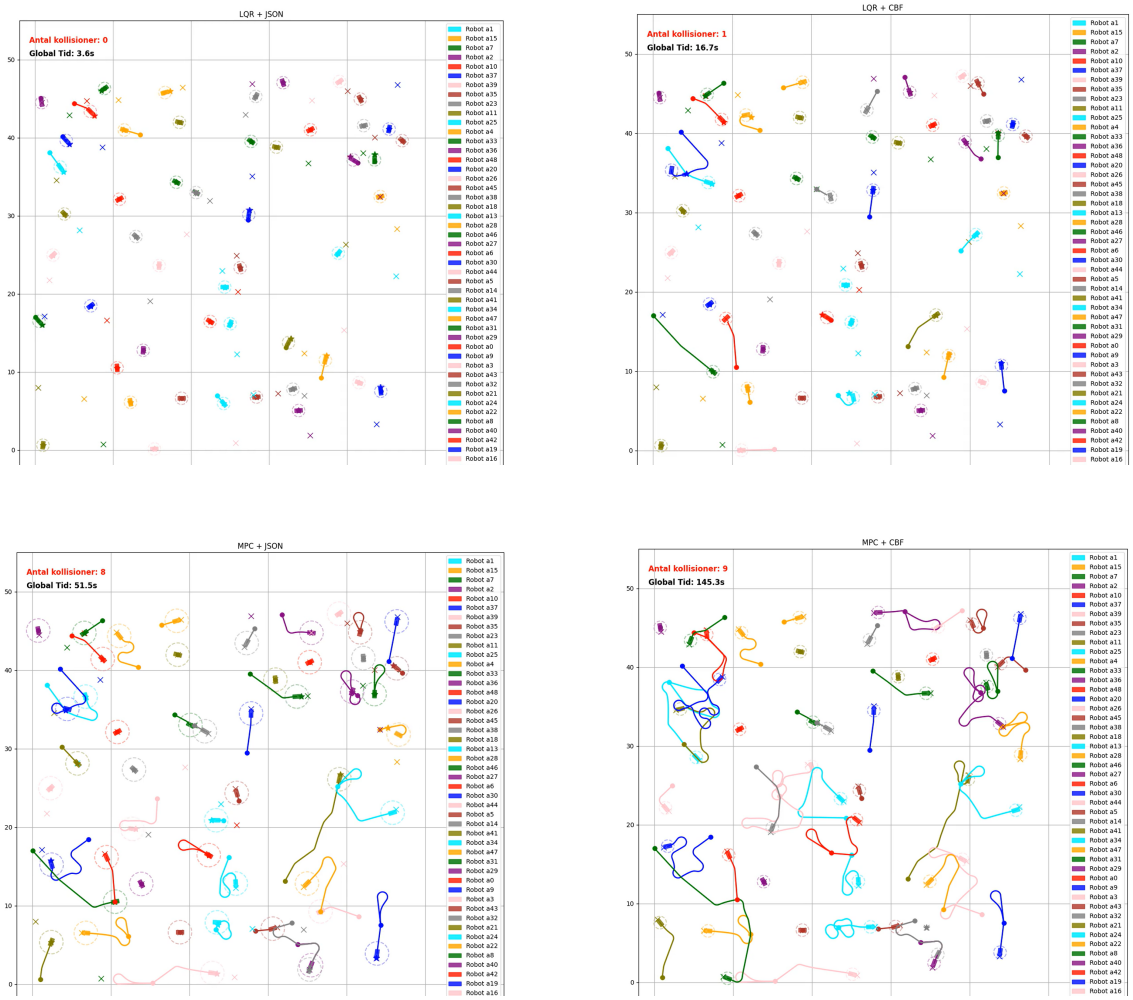
Figur 7.8


LQR + JSON
Antal kollisioner: 0
Global Tid: 7.2s
LQR + CBF
Antal kollisioner: 1
Global Tid: 47.5s
MPC + JSON
Antal kollisioner: 2
Global Tid: 61.9s
MPC + CBF
Antal kollisioner: 1
Global Tid: 149.0s

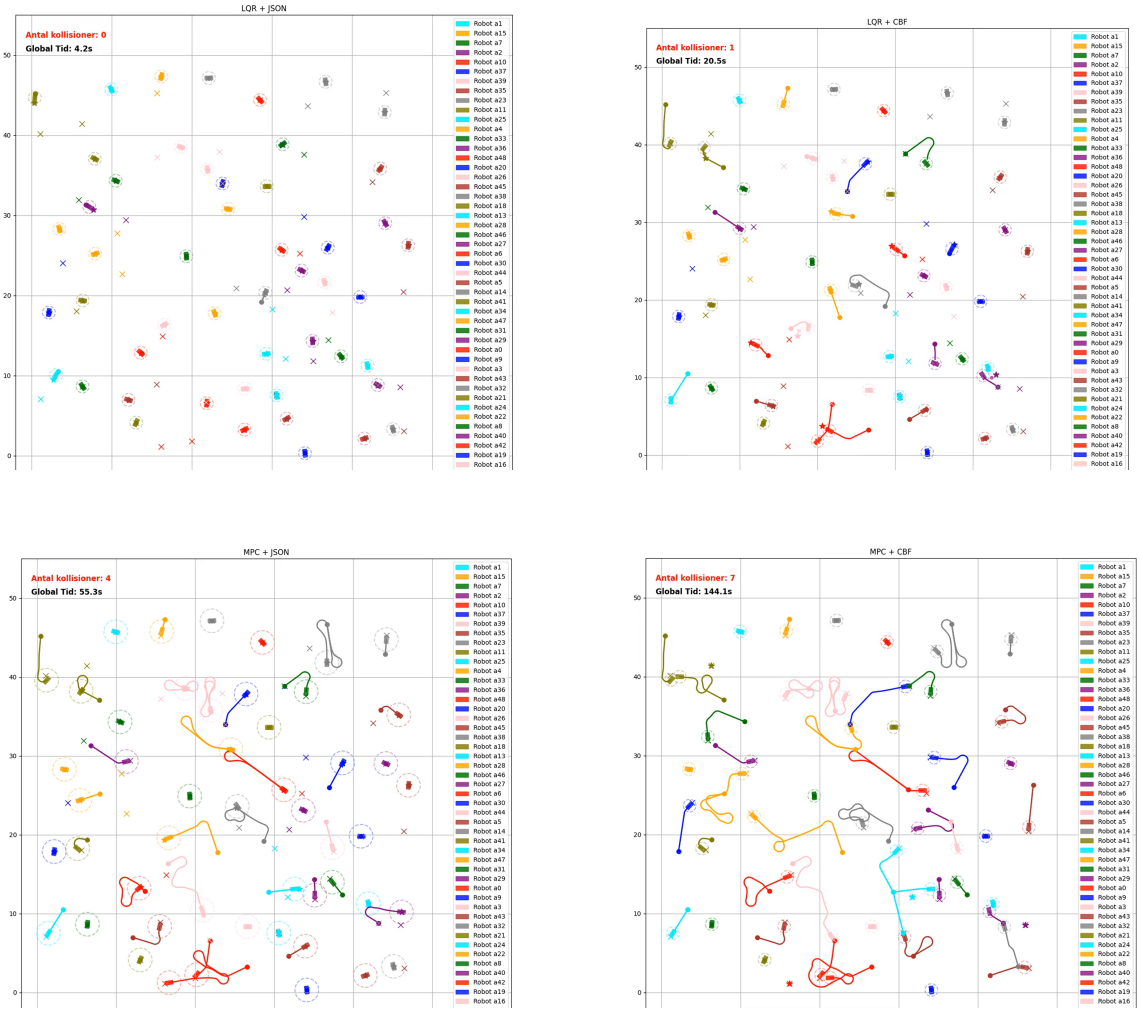
Figur 7.9


LQR + JSON
Antal kollisioner: 0
Global Tid: 4.6s
LQR + CBF
Antal kollisioner: 1
Global Tid: 19.1s
MPC + JSON
Antal kollisioner: 2
Global Tid: 47.8s
MPC + CBF
Antal kollisioner: 5
Global Tid: 140.5s

Figur 7.10


LQR + JSON
Antal kollisioner: 0
Global Tid: 3.6s
LQR + CBF
Antal kollisioner: 1
Global Tid: 16.7s
MPC + JSON
Antal kollisioner: 8
Global Tid: 51.5s
MPC + CBF
Antal kollisioner: 9
Global Tid: 145.3s

Figur 7.11


LQR + JSON
Antal kollisioner: 0
Global Tid: 4.2s
LQR + CBF
Antal kollisioner: 1
Global Tid: 20.5s
MPC + JSON
Antal kollisioner: 4
Global Tid: 55.3s
MPC + CBF
Antal kollisioner: 7
Global Tid: 144.1s

Figur 7.12

7.2 Källkod och GitHub-arkiv

All källkod som har utvecklats och använts för simuleringarna i detta examensarbete, finns öppet tillgänglig på vår GitHub-sida.

Litteratur

[1] A. Combrink, "Advances in Multi-Agent Path Finding: Scalable Lifelong Planning and Optimal Continuous-Time Solutions," Licentiate thesis, Chalmers University of Technology, Gothenburg, Sweden, 2025. URL: https://research.chalmers.se/publication/549458/file/549458_Fulltext.pdf.

[2] Z. Shen, H. Du, L. Yu, W. Zhu och M. Zhu, "A Path Planning and Tracking Control Algorithm for Multi-Autonomous Mobile Robot Systems Based on Spatiotemporal Conflicts and Nonholonomic Constraints," *Actuators*, årg. 13, nr 10, s. 399, 2024. DOI: 10.3390/act13100399.

[3] F. L. Lewis, D. Vrabie och V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

[4] M. Shoaib. "Implementing NMPC with CasADi: Cart-Pole System." Accessed: 2026-05-19. URL: https://medium.com/@shoaib6174/implementing-nmpc-with-casadi-cart-pole-system-62f1d72f2efa.

[5] Virtual Simulation Lab, *Model predictive control Inverted pendulum on a cart using CasADi*, YouTube video, Accessed: 2026-04-28, 2020. URL: https://www.youtube.com/watch?v=NkeqFtKH4Yo.

[6] D. F. Griffiths och D. J. Higham, *Numerical Methods for Ordinary Differntial Equations: Initial Value Problems* (Springer Undergraduate Mathematics Series). Springer, 2010, ISBN: 978-0-85729-147-9. DOI: 10.1007/978-0-85729-148-6.

[7] A. Ravankar, A. A. Ravankar, Y. Kobayashi, Y. Hoshino och C.-C. Peng, "Path Smoothing Techniques in Robot Navigation: State-of-the-Art, Current and Future Challenges," *Sensors*, årg. 18, nr 9, s. 3170, 2018. DOI: 10.3390/s18093170.

[8] A. Khamis, A. Hussein och A. Elmogy, "Multi-robot Task Allocation: A Review of the State-of-the-Art," i *Cooperative Robots and Sensor Networks 2015*, Springer, 2015, s. 31–51. DOI: 10.1007/978-3-319-18299-5_2.

[9] Y. Chen, U. Rosolia och A. D. Ames, "Decentralized Task and Path Planning for Multi-Robot Systems," *IEEE Robotics and Automation Letters*, årg. 6, nr 3, s. 4337–4344, 2021. DOI: 10.1109/LRA.2021.3068103.

[10] M. Jankovic och M. Santillo, "Collision Avoidance and Liveness of Multi-agent Systems with CBF-based Controllers," i *2021 60th IEEE Conference on Decision and Control (CDC)*, IEEE, 2021, s. 6822–6828. DOI: 10.1109/CDC45484.2021.9682854.

[11] S. Zhao, Z. Yan, T. Huang och S. Wen, "A Comprehensive Review on Control Barrier Functions: Uncertainty Handling, Design Optimization, and Feasibility

Analysis," *IEEE Transactions on Cybernetics*, årg. 56, nr 4, s. 2061–2069, 2026. DOI: 10.1109/TCYB.2025.3633800.

[12] M. Schwenzer, M. Ay, T. Bergs och D. Abel, "Review on model predictive control: an engineering perspective," *The International Journal of Advanced Manufacturing Technology*, årg. 117, nr 5–6, s. 1327–1349, 2021. DOI: 10.1007/s00170-021-07682-3.

[13] J. Breeden och D. Panagou, "Compositions of Multiple Control Barrier Functions Under Input Constraints," i *2023 American Control Conference (ACC)*, IEEE, 2023, s. 3688–3695. DOI: 10.23919/ACC55779.2023.10156145.

[14] COIN-OR, *Ipopt Documentation*, https://coin-or.github.io/Ipopt/, Accessed: 2026-05-25, 2026.

[15] NumPy Developers, *NumPy News*, https://numpy.org/news/, Accessed: 2026-05-25, 2026.

[16] Matplotlib Developers, *Matplotlib: Visualization with Python*, https://matplotlib.org/, Accessed: 2026-05-25, 2026.

INSTITUTIONEN FÖR ELEKTROTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se

CHALMERS