Effective robot arm motions

stability and efficiency through natural dynamics

Wolfslag, Wouter

DOI
10.4233/uuid:a43c0c07-5908-4890-99b8-1744194f815b

Publication date
2018

Document Version
Final published version

Citation (APA)
Wolfslag, W. (2018). *Effective robot arm motions: stability and efficiency through natural dynamics*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:a43c0c07-5908-4890-99b8-1744194f815b

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Effective robot arm motions

Stability and efficiency through natural dynamics

Wouter Wolfslag

Effective robot arm motions

stability and efficiency through natural dynamics

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus, Prof.dr.ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties
in het openbaar te verdedigen op
donderdag 25 oktober 2018 om 12:30 uur

door

Wouter Jan WOLFSLAG
Werktuigkundig ingenieur
Technische Universiteit Delft, Nederland
geboren te Leiden

Dit proefschrift is goedgekeurd door de promotoren.

Samenstelling van de promotiecommissie:

Rector Magnificus	voorzitter
Prof. dr. ir. M. Wisse	Technische Universiteit Delft, promotor
Prof. dr. R. Babuška	Technische Universiteit Delft, promotor

Onafhankelijke leden:

dr. N. Mansard	LAAS/CNRS, Frankrijk
dr. Y. Tassa	Google DeepMind, Verenigd Koninkrijk
Prof. dr. H. Bruyninckx	Katholieke Universiteit Leuven, België
Prof. dr. A.J. van der Schaft	Rijksuniversiteit Groningen
Prof. dr. Ing. H. Vallery	Technische Universiteit Delft
Prof. dr. J.W. van Wingerden	Technische Universiteit Delft, reservelid

Dr. Ir. M.C. Plooij heeft een grote bijdrage geleverd aan de totstandkoming van dit proefschrift.

Dit onderzoek is financieel ondersteund door de Technologiestichting STW (project nummer 11282).

ISBN 978-94-028-1221-3

'It ain't what you do, it's the way that you do it. That's what gets results.'

- Melvin Oliver/James Young, 1939

Contents

Summary

Effective robot arm motions

stability and efficiency through natural dynamics

While progress in many fields of robotics has been swift, robot arm movement in scenarios without contact has changed little in the last decades. This lack of change is not due a lack of potential for improvement. After all, the human arms that these robot emulate move in ways that are more robust, energy efficient and adaptable. This thesis is inspired by human movement skill in these three aspects to improve the movement of robotic arms in non-contact situations. The six main contributions of this thesis are divided over those three aspects. The aspects are studied for two motions, the reaching motion, that is, move from the initial position to a pre-specified target position and back, and the pick-and-place motion, which adds picking and placing an object at the initial and target positions respectively.

The first aspect, robustness, is studied from a stability standpoint. The first aspect is the topic of the first part of this thesis, which contains four of its six main contributions. Stability, as understood in this thesis, is the property of returning to a fixed (desired) motion after an initial disturbance two motions. This form of stability is a minimal requirement for successful task completion. Most current robot arms rely on fast sensory feedback for their stability. This contrasts with humans, who rely on skill at choosing motions that are intrinsically stable. Such self-stable motions have been used by earlier robotics researchers to make robots that juggle or walk without the need for sensory feedback. However, these robots perform tasks involving impacts, which can have a large stabilizing effect. No such impacts are available in reaching motions. A self-stabilizing reaching motion instead depends on ingenious use potential energy and centrifugal or Coriolis effects.

The first contribution of this thesis is to show in simulation that self-stabilizing motions for a two degree of freedom robot arm performing a reaching task do exist. The arm has two properties that are required for stabilizing the motion: a spring on the first joint, and mechanical friction in both joints. These two properties are not sufficient, as the stability of the motion depends on energy transfer via convective acceleration terms to stabilize the second joint. When the feedforward input corresponding to a stable reaching motion was applied to a physical robot arm, it was found to be stable as well.

While the resulting motion is stable, it does not reach the desired end-positions, due to difference between the simulation and the physical robot. In investigations I was part of but were not included in this thesis, it was shown that it is sometimes

possible to find motions which are insensitive to these model-reality differences. These motions do not account for all types of model-reality differences and also do not consider stability. These results suggest self-stabilization alone is not sufficient to make a robot arm perform an accurate reaching task.

Yet, the second contribution of this thesis is to perform a precise reaching motion with a controller that almost exclusively depends on self-stabilization. Almost exclusively, because the proposed control scheme consists of two phases of which the first one uses some feedback, in the form of repetitive control, which is similar to iterative learning control. The feedback in repetitive control has a delay of one cycle of the reaching motion, and is used to learn the feedforward input that tracks the desired motion. In the second phase, the system exclusively relies on self-stabilizing effects. I show that these effects depend only on the motion of the system, and not on the input needed to track that motion.

The third contribution of this thesis with respect to self-stability is to show that this two phase approach can also be used to perform a pick-and-place task, rather than the reaching task investigated above. The robot arm uses self-stabilization to blindly perform this pick-and-place task, after an initial learning phase.

For the fourth contribution, the last on self-stability, I investigated the use of optimizing feedforward controller (motion) for self-stability in situations in which feedback is available. I show that such feedforward controllers, when compared with unsuitable feedforward controllers, make a significant difference in the accuracy of a robot arm performing a reaching task. However, when comparing feedforward controllers optimized for self-stability with standard feedforward signals, the difference is only meaningful when the feedback uses small gains.

The second aspect studied in this thesis is energy efficiency. It is inspired by the efficiency of human movement, as enabled by muscles. Many previous researchers were inspired by muscles, most notably by their elastic properties. In this thesis I am inspired by muscles using a different mechanism to effectively perform negative work. The research in this aspect therefore focusses on the use of brakes.

The fifth contribution is to pose the design of a plugless robot arm: a robot arm that performs a pick-and-place task where the pick position is higher than the place position. The drop in potential energy of the payload can be used to power the robot arm, hence the name plugless arm. The main task of such an arm is to steer the end-effector to the right place position, a task that in principle can be executed without the motor performing positive work. A side effect is that the motor could thus be replaced by mechanically simpler brakes. This thesis investigates the possibility of controlling a robot arm with only brakes, and shows three different approaches to achieve that task.

The third inspiring aspect of human movement that is studied in this thesis is its adaptability. For robot motions without impact, adaptability comes from being able to quickly plan and replan motions. Furthermore, this planning should account for dynamic quantities such as velocity and torque constraints.

Three techniques were combined in order to plan faster: optimal control, supervised learning and sampling-based planning. The latter handles obstacles, and splits the whole motion planning problem into multiple smaller planning problems that are easier to solve. Optimal control can handle those smaller planning problems, but is too slow. Supervised learning generalizes input-output relations from a dataset, and provides fast output after learning. By learning from a dataset of precomputed optimal trajectories, the computation time is shifted offline, making the planning itself much faster.

The last contribution, and the only one related to adaptability, is to improve the state-of-the-art technique for learning in sampling-based planning. The improvement comes from picking a more suitable optimal control approach. This classic approach, called indirect optimal control, is not used often any more because of poor numerical performance in many practical settings. For use in a sampling-based planner, it carries two benefits. First, the steering input is parameterized using a small number of parameters, which makes it easier to learn this input. Second, the dataset of optimal trajectories can be constructed without numerical optimization. These benefits lead to tenfold speedup in both the offline trajectory generation phase and the online planning phase.

In conclusion, for this thesis I studied robot arm movement inspired by three aspects of human motion. First, the self-stability of human movement inspired me to make a feedforward controlled robot arm perform a pick-and-place task. Second, human strategies for energy efficient movements caused me to design a controller for a robot arm only actuated by brakes. Finally, I aimed for more adaptive robot movement by improving a sampling-based trajectory planning algorithm. This improvement led to a tenfold reduction of planning times.

Samenvatting

Effectieve bewegingen voor robotarmen

stabiliteit en efficiëntie door natuurlijke dynamica

Hoewel een groot deel van de robotica snel vooruit is gegaan zijn de bewegingen van robotarmen, in situaties waarin de robot geen contact maakt met de omgeving, in de laatste decennia weinig veranderd. Dit gebrek aan verandering komt niet door een gebrek aan potentie voor verbetering. Menselijke armen, die de inspiratie voor deze robots vormen, bewegen tenslotte op een manier die leidt tot robuustheid, een hoog (energie) rendement en een groot aanpassingsvermogen. Deze thesis is geïnspireerd op menselijke vaardigheden in juist deze drie aspecten, met als doel robotarm-bewegingen zonder contact te verbeteren. De zes belangrijkste bijdragen van deze thesis zijn verdeeld over deze drie aspecten. Deze aspecten bestudeer ik voor twee bewegingen: het reiken en het pakken. Reiken is het heen en weer bewegen naar een vooraf bepaalde positie, pakken is dezelfde beweging waarbij ook nog een object wordt opgepakt en neergelegd.

Het eerste aspect dat bekeken wordt is robuustheid, specifiek de robuustheid van stabiliteit. Stabiliteit, zoals het in deze thesis bedoeld wordt, is de eigenschap dat de robot terugkeert naar een vaste (gewenste) beweging na een initiële verstoring. Deze vorm van stabiliteit is een minimale eis om een taak te kunnen volbrengen. De meeste huidige robotarmen gebruiken snelle sensorfeedback om zichzelf te stabiliseren. Mensen daarentegen hebben de vaardigheid om bewegingen te kiezen die uit zichzelf stabiel zijn, en dus geen feedback nodig hebben. Binnen de robotica zijn zelf-stabiliserende bewegingen al gebruikt om robots te laten jongleren of lopen zonder gebruik te maken van feedback. Deze zelf-stabiliserende robots maakten gebruik van botsingen in het contact met de bal of de grond. Deze botsingen kunnen stabiliserend werken, maar komen niet voor in reikbewegingen. Een zelf-stabiliserende reikbeweging moet gebruik maken van natuurlijke dynamica, dat wil zeggen de potentiële energie, en de centrifugaal- of Coriolis effecten die in het ontwerp van de robotarm zijn opgesloten.

De eerste bijdrage die deze thesis levert, is het in simulatie laten zien dat zelf-stabiliserende reikbewegingen voor een robotarm met twee graden van vrijheid bestaan. De bestudeerde robotarm heeft twee eigenschappen die nodig zijn voor het bestaan van zulke bewegingen: een potentiële energie (veer) aan het eerste gewricht en wrijving in beide gewrichten. Deze twee eigenschappen zijn echter niet voldoende: de stabiliteit van de beweging hangt af van centrifugaal- en Coriolis effecten. De input die de beweging in simulatie veroorzaakt blijkt bij tests op een fysieke robotarm ook een stabiele beweging te veroorzaken.

Hoewel de resulterende beweging op de fysieke arm stabiel is, bereikt deze niet de gewenste eindposities. Dit komt door een verschil tussen de simulatie en de fysieke robot. In onderzoek waarin ik een bijdrage heb geleverd, maar dat niet in deze thesis is opgenomen, is laten zien dat het soms mogelijk is bewegingen te vinden die niet gevoelig zijn voor dit verschil tussen model en werkelijkheid. Het bleek echter dat die bewegingen niet alle vormen van model-onnauwkeurigheid kunnen tegengaan. Daarbij hielden deze bewegingen ook geen rekening met stabiliteit. Dit maakt het onwaarschijnlijk dat het mogelijk is een input te vinden die op een fysieke robotarm een reikbeweging stabiel en nauwkeurig uitvoert.

Toch is de tweede bijdrage van deze thesis het nauwkeurig uitvoeren van een reikbeweging die bijna volledig van zelf-stabilisatie afhangt. Bijna volledig, want de voorgestelde regelaar bestaat uit twee fases, waarvan de eerste fase enige feedback gebruikt, in de vorm van Repetitive Control, een variant van Iterative Learning Control. De feedback in Repetitive Control is met een bewegingscyclus vertraagd, en wordt gebruikt om de input te leren die de gewenste beweging volgt. In de tweede fase is het systeem volledig afhankelijk van de zelf-stabiliserende effecten in die beweging. Ik laat zien dat deze effecten alleen afhankelijk zijn van de beweging, en niet van de input die nodig is om de beweging te volgen.

De derde bijdrage is het laten zien dat deze twee-fase regeling ook gebruikt kan worden als de robot objecten pakt. Na een initiële inleerperiode is de robotarm in staat om blind objecten tussen twee posities te verplaatsen.

Voor de vierde bijdrage, de laatste op het aspect van zelf-stabiliteit, heb ik bewegingen die geoptimaliseerd zijn voor zelf-stabiliteit onderzocht in situaties waarin feedback ook beschikbaar is. Deze geoptimaliseerde bewegingen blijken de robotarm een reiktaak nauwkeuriger te laten uitvoeren dan minder geschikte bewegingen. Echter, in vergelijking met standaard gebruikte bewegingen leidt het optimaliseren van de bewegingen voor zelf-stabiliteit alleen tot een betekenisvol verschil als de gebruikte feedback acties klein zijn.

Het tweede aspect van menselijke bewegingen dat de inspiratie vormt voor deze thesis is energie efficiëntie. Eerder onderzoek op dit gebied werd geïnspireerd door de elasticiteit in menselijke spieren. In deze thesis wordt de inspiratie gehaald uit de verschillende mechanismes die spieren gebruiken om positieve en negatieve arbeid te leveren. Voor dit aspect wordt dan ook onderzoek gedaan naar het gebruik van remmen.

De vijfde bijdrage is het voorstellen van de stekkerloze arm, een robot die een pick-and-place taak uitvoert waarbij de pak-positie hoger is dan de plaats-positie. Het verschil in potentiële energie kan gebruikt worden om de robotarm van vermogen te voorzien, zodat er geen stekker meer nodig is. Het sturen van deze robotarm

kan in principe gedaan worden zonder dat de motoren positieve arbeid hoeven te leveren. Hierdoor kunnen de motoren vervangen worden door, mechanisch simpelere, remmen. Deze thesis onderzoekt de mogelijkheid om de robot aan te sturen met alleen maar remmen, en laat drie manieren zien om dit te doen.

Het derde aspect dat onderzocht wordt is het aanpassingsvermogen voor bewegingen. Aanpassingsvermogen komt voort uit het snel kunnen plannen en aanpassen van bewegingen. Bovendien moet die planning rekening houden met dynamische aspecten zoals snelheid en limieten van de motoren.

Drie technieken worden gebruikt om deze bewegingen snel te plannen: Optimal Control, Supervised Learning en Sampling-Based Planning. Sampling-Based Planning houdt rekening met obstakels en splitst het plannen van grote bewegingen op in het plannen van meerdere kleine bewegingen, die makkelijker te vinden zijn. Optimal Control kan gebruikt worden om deze kleine bewegingen te vinden, maar is te langzaam. Supervised Learning generaliseert de input-output relaties in een dataset, en werkt snel na het leren. Door te leren van een dataset van met Optimal Control gevonden bewegingen, wordt de rekentijd naar een offline fase gebracht, waardoor de online snelheid van het plannen verhoogd wordt.

De zesde en laatste bijdrage is een betere techniek voor het leren binnen Sampling-Based Planning. De verbetering komt voort uit het kiezen van een geschiktere vorm van Optimal Control. Het blijkt dat het zogenoemde Indirect Optimal Control, een klassieke methode die voor veel toepassingen gedateerd is, toch twee voordelen heeft binnen Sampling-Based Planning. Ten eerste wordt de beweging door een klein aantal parameters beschreven, wat het leren vereenvoudigd. Ten tweede kan de dataset gegenereerd worden zonder numerieke optimalisatie. Deze voordelen leiden tot zowel een snellere offline fase als een snellere planning fase.

Concluderend, in deze thesis heb ik robotarm-bewegingen bestudeerd geïnspireerd door drie aspecten van menselijke bewegingen: zelf-stabiliteit, energie-efficiëntie en aanpassingsvermogen. Eerst heeft de zelf-stabiliteit van menselijke bewegingen mij geïnspireerd tot het maken van een robotarm die op basis van zelf-stabiliteit een pick-and-place taak uitvoert. Ten tweede heeft het voorbeeld van menselijke energie efficiëntie geleid tot het onderzoeken van het aansturen van een robotarm met alleen maar remmen. Ten slotte heb ik, als een stap richting het bereiken van een menselijk niveau van aanpassingsvermogen, een Sampling-Based plan-algoritme verbeterd. Deze verbetering leidde tot een tienvoudige vermindering van de rekentijd voor het plannen.

1

Introduction

This thesis presents a study into the movement of industrial robot arms. Inspired by human arms, I identified five aspects of these arms that can be improved upon, as detailed in Section 1.1. Two of these aspects consider contact with other objects, a field that is already extensively studied. The remaining three, which do not consider contact, are studied in this thesis from a technical viewpoint. They are: stability by using natural dynamics, energy efficiency from intelligently applied actuation and adaptability through fast path planning in state space. While the way robots execute non-contact motions has remained almost unchanged for 50 years, I believe that there is much potential for better performance by looking at these aspects. Table 1.1 shows these three topics, the chapters in which they are investigated, and the techniques used in these chapters.

Table 1.1: Topics in this thesis, and the techniques used for them.

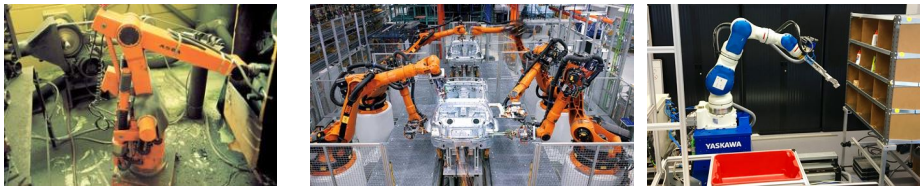
Topic	Chapters	Techniques
Self stability	2-5	Direct optimal control, LMIs, Iterative learning control, Gait sensitivity norm, Limit cycle stability
Energy efficiency	6-7	MCTS, Non-linear control, Reinforcement learning
Trajectory planning	8	Indirect optimal control, RRT, Supervised learning

1.1 | Motivation

Within the field of robotics, industrial robot arms are a key area for study due to the amount of potential applications in society. They promise fast adaptability in industrial automation, thereby making automation accessible to small and medium sized enterprises. In a quest to fulfil that promise, various groups throughout Europe pursue research on industrial robot arms [2, 19, 188] and their software [73, 105]. The Delft Biorobotics lab is a part of this research, for instance, through the Factory in a Day project [31], involvement in ROS Industrial [63] and by winning the Amazon Robotics challenge [64]. This thesis is a part of that effort.

One would expect robot arms to have changed rapidly, especially given the amount of research devoted to them. Surprisingly, the opposite is true, as industrial robot arms have changed surprisingly little over the last 50 years. This can be seen in Figure 1.1, which shows the first microcomputer-controlled robot, a modern car factory with welding robots, and the robot that was used by the Delft team in the Amazon Robotics challenge 2016. This lack of change is observable in robot control as well. While the exact controllers are kept secret, most industrial robots today stabilize themselves around preplanned motions.

While modern robot arms are similar to arms from 50 years ago, they differ fundamentally from human arms. This is surprising, as the robots were developed



(a) **(b)** **(c)**

Figure 1.1: Past and current industrial robots: (a) The ABB IRB 6 from 1974, the first microcomputer controlled robot, (b) A modern robot arm factory, and (c) the robot of the TU Delft team for the Amazon Robotics challenge, 2016. For attribution see references [1], [182] and [43] respectively.

to mimic, and hopefully improve upon, human arms. Current robot arms can lift heavier weights, move more repeatably and more precisely than humans do. Still, the natural and proficient way humans move is still a league above the best moving robots. Inspired by human motion, I identified five aspects of interest.

The first two aspects consider tasks in which contact plays an important role.

- Sensing capability: sensors in human arms detect orientation and contact. In particular, our skin can sense normal forces, sliding and roughness of the contact surface, among other things [37, 81].
- The ability to handle collisions: next to being able to sense contact, human arms are also highly adapted to contact. In everyday life, humans allow contact everywhere along their arms, using it to manipulate objects or stabilize themselves.

The last three aspects of human motion that are inspiring for robot arms involve non-contact tasks.

- Self stability: humans select motions that minimize the task-relevant sensitivity to noise [88], likely to cope with noisy and delayed sensors and actuators. Furthermore, in the related field of locomotion, it has been shown that human-like motions emerge when searching for self-stabilizing motions [34].
- Energy efficiency: human arms are more energy efficient, thanks to the way they are actuated. For instance, the intrinsic properties of human muscles allow effective energy dissipation [7]. Furthermore, due to bi-articular muscles, humans can transfer energy between the links of the arm, which is more effective than dissipating and injecting energy at the same time, as robots do [74].

- Adaptive planning: humans can quickly adapt their behavior to different situations. For arms, this means the capability to move swiftly while not hitting obstacles in unfamiliar environments, and handling new tools.

Contact tasks have been an important topic of research for decades [39, 140]. In particularly, in the last decade this research focused on three aspects: 1) handling of contact for control via planning through contact [87, 115], 2) stiff joint control using position, force and impedance [3, 25] or by using compliant designs [138], and 3) the capability robot hands for in-hand manipulation [8, 186]; and into developing sensor skins to detect (impending) contact [188].

At the same time, very little attention has been paid to the non-contact aspects of robot arm motion, despite it being clear that the three listed aspects can potentially greatly improve robot arm performance. That is why this thesis studies the non-contact aspects: self-stability, energy efficiency and adaptive planning.

While these three aspects seem only loosely related, and while they will lead to separate research questions, they are coherent in a key element, as I will approach them from an optimal control/trajectory planning perspective. That is, the way to pick the trajectory is the central question throughout this thesis. This approach makes it possible to draw upon the past experience in the lab on dynamically challenging tasks such as energy-efficient bipedal locomotion.

To study robots arms, I focus on the reaching task, moving the end effector from one known position to another. This task is performed on systems with one or two degrees of freedom, which were chosen as the simplest systems that could show off the concepts under study. This task is relevant for almost all robot arms, although no attempt is made to bridge the gap to full-scale industrial arms. The main advantage of this task is that it allows us to focus on the motions of the robot by excluding or greatly simplifying other factors that are also important for robot arms in general, such as higher level decision making, object recognition and sensor calibration.

The following sections discuss the three inspirational aspects seperately. They give a more in-depth overview of the research fields in robotics that are related to each aspect, and lead to the research questions and goals tackled in this thesis. For an immediate overview of those research questions and goals, refer to Section 1.5.

1.2 | Self stability

The first striking aspect of the way humans move is their skill at using self-stabilizing effects. By self-stabilizing effects, I mean effects in the dynamics of

the system that cause certain trajectories of the system to be stable. In this thesis, the term stability is used to describe the property called asymptotic stability, see [145] for a technical definition. The practical relevance of this type of stability is that it demands that motions converge towards a desired trajectory, independent of the starting condition. This means that the system can recover from an initial disturbance.

By picking the right motor signal, stability emerges, seemingly without effort, due to the self-stabilizing effects in the natural dynamics of the arm. The skill of finding these motions allows humans to perform tasks within a system whose motions generally are not stable, and to do so without depending on a feedback controller. Of course, human movement is not only dependent on self-stabilization. However, it is an important factor, especially for fast motions, for which humans cannot rely on feedback at all, due to large delays (order of 100 ms [110, 161, 170]). The delays are partially counteracted by using an internal model, that predicts behavior [42]. Still, this internal model makes errors, and it is likely that humans look for motions that minimize this error [51] or at least the variability in the resulting motion [10].

Motivated by human self-stability and its perceived benefits, such as energy efficiency, researchers have tried to make robots use self-stabilizing effects. The most famous examples are (nearly) passive bipedal walkers [34] and juggling robots [146].

It is remarkable that the use of self-stability has received virtually no research attention for industrial robot arms. This lack of attention is despite potential benefits like energy-efficiency and robustness against sensor delays, noise or failure.

The question arises whether robot arms can even use the self-stabilization approach when performing tasks. The examples mentioned above all rely on impacts to help stabilize the system. While, for example, the oscillating inverted pendulum [111, 159], uses non-impact dynamics for stabilization, it does nothing apart from stabilization. Therefore the overarching question for the chapters in this thesis inspired by self-stability is *Is it possible to make a self-stabilizing robot arm perform a pick-and-place task?* This question will be studied in three steps.

Robot arms can be stabilized in three different ways, or a combination thereof. In most robots, stability is obtained via feedback: sensors measure the robots position, and an on-board computer adjust the torque commands to the actuators in order to recover. As robots are often complex multi-dof systems, choosing the right torque commands can be a challenging task. Various schemes, e.g. sliding mode, feedback linearization, backstepping [145], have been devised. Most of these work by suppressing the natural dynamics, thereby avoiding possibly negative influences of, for instance, the Coriolis or centrifugal effects. However, by

suppressing these effects, their positive influences are also discarded. Two approaches aim to address that issue: passivity-based control and model predictive control. The passivity-based approaches use system properties to find effects that do not need to be cancelled out [45, 121, 148]. Model predictive control computes the optimal feedback controller [40, 86], often with model simplifications such as a limited time horizon or linearized dynamics. In all cases, the controllers rely on the availability of a timely, reliable feedback signal, [91].

The second approach is the simplest way to stabilize a robot without feedback, namely to use mechanical constraints, such as end stops. This approach is frequently used, for instance, during a homing phase during the start up of a robot with relative angle sensing. The use of mechanical constraints has been extended to allow robots to align objects without knowing their initial position [48]. The constraint approach however is limited, because even small deviations in the task of a robot require the design of the robot to be adjusted.

The last approach is to the interplay between the natural dynamics of the system, and the trajectory being followed. The goal is to find a trajectory that uses the natural dynamics to ensure that the system is stable around that trajectory being followed. This approach does not rely on feedback, and can handle deviations in a robots task without adjusting the robots design. That makes the last approach interesting to study.

To highlight the difference between the feedback control approach and this trajectory selection approach, I use the term feedforward control for the latter. For the fully actuated robot studied, the trajectory can be regarded as a function of a nominal initial state and an input signal, the feedforward controller. The inverse of this statement is also true, justifying the interchangeable use in this thesis of the terms feedforward controller design and trajectory selection. Other uses of the term feedforward controller, such as terms in a feedback controller that do not directly use an error signal, are avoided.

A question that immediately arises when pursuing a purely feedforward control approach is how to assess the stability of practically useful robot arm motions. The stability of a non-linear system around a desired trajectory cannot be assessed for many trajectories. There are two important classes of trajectories for which it can be. First, those trajectories that converge towards a steady state. In [106] the stability of such trajectories is analyzed for port-Hamiltonian systems, a wide class of systems that includes serial chain manipulators. Unfortunately, steady state analysis is only sufficient if the robot is allowed to stay in the same state for an extended amount of time, while robots are typically asked to start moving again shortly after having reached a desired state. The second class of trajectories is more relevant to the case of robot arms. This class consist of cyclic or periodic

trajectories. Robots are often used to perform repetitive tasks, which can be executed as periodic motions. Furthermore, periodic motions allow stability analysis along the same lines as for limit cycles [104, 162].

Limit cycle stability has been used to perform juggling [146] and, perhaps its most prominent successful application, bipedal walking. First, McGeer et al. [109] showed that some legged devices are stable when walking down a slope, even though they are not stable when standing still. His approach was followed in various research that used limit cycles as a basis of designing controllers for bipedal robots [34]. While these robots are equipped with sensors, they rely on impact for stabilization. So although their controllers use some feedback, they still indicate the potential of limit cycles in a sensorless setting. The first step in answering the main question on self-stabilizing robot arms is *to show that feedforward control by itself can make robot arms exhibit a stable limit cycle.*

It must be noted, that the stability analysis is offline and model based. This means, that there will inevitably be significant differences between the analyzed system and the robot. For some motions, such as walking, minor deviations will alter the motion, but keep a comparable performance [67]. The reaching task is not such a motion. Errors in positioning larger than a centimeter or so, almost always mean that the task is not completed. Therefore, the robot feedforward controller must be robust, i.e., able to counteract the effects of model-robot mismatches.

The effects of model-robot mismatches was investigated in [14] for the context of differential drive robots. They showed it is possible to steer such a robot to a desired position, even when the wheel diameter is uncertain. Their approach centers on the idea that the effects of the uncertainty cancel out for some specific trajectories. In research not included in this thesis, I followed that idea to show that for some uncertainties in a robot arm model, such as an unknown Coulomb friction coefficient, it is possible to find paths that allow accurate positioning of the robot [128]. It was also shown, that some uncertainties do not allow for accurate positioning using feedforward control only. Furthermore, the approach taken does not account for the stability of the motion.

As a result, a sensorless robot arm that is both self-stabilizing and robust is unlikely to exist. A more promising way to handle the model-robot mismatch is via learning. While learning is based on sensory feedback, it poses little demands on this feedback. For instance, the feedback does not need to be real time, or data can be missing. We can expect the robot to be self-stabilizing and robust without the data, after a learning period. The second step in answering the main question on self-stabilizing robot arms uses this approach *to make a robustly stable robot arm, such that a robot with limited sensing can learn to perform a self-stabilizing motion, despite both model uncertainty and disturbances.* To achieve this goal, a

method is developed to verify the robust stability of the trajectory of the robot arm. It is shown that the stability of this trajectory does not depend on the feedforward controller needed to achieve this trajectory. The feedforward controller can thus be learned, which is done using a standard Repetitive Control scheme, a variant of Iterative Learning Control [98]. The experimental goal of performing a pick and place task is reached in two stages: the first is to develop the method to perform robustly stable reaching motions, the second is *to show the two staged method can be used to perform a pick-and-place task.*

In the first two steps of the self-stability part of this thesis, the capacity of robot arms to self-stabilize was investigated and improved. This started from the observation that self-stabilizing behavior is clearly advantageous for humans. However, humans and robots are not built the same way, so do not necessarily have to be controlled the same way either. Past research seems to confirm this, as the use of self-stabilizing effects have resulted in very energy efficient robot walkers [34], but it has only resulted in reliable gaits for one specific task: walking on level ground [18]. So, after developing the methods to robustly self-stabilize a robot arm, the third step looks at the applicability of these methods, by studying the following question: *is optimizing the feedforward controller for self-stability a useful approach to improving the accuracy of robot arm motions?*

1.3 | Actuation and energy consumption

The brains and muscles of humans have evolved to be a very energy efficient actuation system. The second part of this thesis is inspired by this energy efficiency, and investigates how to use properties of human muscles to obtain it for robot arms. I am not the first to be inspired by human muscles. In fact, it is even possible to identify three, partially overlapping, branches of robotics research that aim to mimic some properties of muscles.

The first branch concerns the elasticity of muscles. In the famous Hill-type muscle model [183], two types of elasticity are identified: series elasticity and parallel elasticity. The former has the elastic element in parallel with the active, work-producing, element; the latter has it in series.

In robotic actuation both types of elasticity have been used. Series elastic actuators were used for their benefits in accurate torque control, their ability to improve safety and shock tolerance by decoupling the reflected inertia of the motor from the inertia of the link [137]. Furthermore, the same decoupling also allows a series elastic actuator to effect an impedance controller when the elasticity can be modulated [173]. Finally, series elastic actuators can reduce the peak power required

by the motor, allowing the use of lighter motors [123]. Parallel elasticity is applied for taking over part of the torque demands on the motor, thereby decreasing energy loss due to motor heating. The most prominent example is gravity compensation [174, 177, 181]. The characteristics of the parallel elasticity can also be tuned for a specific task, for example [131]. Recent results show how to do so for optimal energy efficiency [160].

Elasticity in actuators is a well-developed research field, with many active contributors. Therefore, while the elasticity in the actuation of the robots is used in this thesis, it is not the main focus of this research.

The second branch of muscle inspiration for robotic actuators is a more global property of some type of muscles: they are multiarticular, that is, they actuate more than one joint at the same time. As such, they empower humans to more effectively transfer energy between links of their extremities [139], avoid cocontraction to produce a desired end effector force direction [149], avoid performing negative work on one joint, while simultaneously performing positive work on another joint [179]. Especially the last property is inspiring for use in robot arms.

In literature, various biarticular robot actuators were proposed. The paper [74] gives a good overview of the different designs. That paper also shows that multiarticular muscles are an active research field, that is not yet well developed. However, as development in this field relies heavily on novel mechanical design, it is outside the scope of this thesis, which focuses on the interaction between control and dynamics.

The third branch of research aimed at incorporating muscle properties in the actuation of robots looks at the capability of humans to effectively do negative work. Human muscles use a different (chemical) mechanism for performing negative work than they do for performing positive work [7, 65]. This suggests using a similar separation for robots. This idea is the least researched of the three branches, and therefore the main inspiration for the work in this thesis.

The dissipative actions can be enacted by brakes. In robot literature, brakes have mainly been used for locking, rather than dissipating energy. Locking allows switching between different actuator modes, such as different transmission ratios [118] or elasticity characteristics [107]. Most prominently, Collins et al. [35] relied on clutches in their design of the first exoskeleton that increased the metabolic efficiency of walking. Recently, research in the use of locking mechanisms has expanded, see [129, 134] for an overview.

In this thesis I focus on the use of dissipative actions, to study their effectiveness. In principle, this comes in two parts: when only brakes are available, and when they are used together with an energy supply (motor). For two reasons, I pick

the former. First, by putting such an extreme limit on the actuators, I hope to learn about the mechanisms underlying energy efficient movement. After all, if you cannot inject energy into the system, you have to be very careful with the energy available. Second, a system that is actuated by brakes only is in some sense intrinsically safe. This is the reason such systems were studied as haptic interfaces before [108, 143, 165]. Such intrinsic safety could also be beneficial for robotic arms.

This leads to the research question that was inspired by human actuation and energy efficiency: *can robot arms perform reaching motions with only dissipative actions?* For this purpose, a robot arm whose actuators do not inject energy in the system is controlled such that it performs a pick-and-place motion.

1.4 | Adaptive planning

Humans also outperform robot arms in adaptability. In non-contact situations, the adaptability of a robot comes from its ability to quickly replan motions. The planning of dynamic motions is done successfully using optimal (feedforward) control to answer the first research questions of this thesis. Numerous well-tested optimal control algorithms are available, see [142] for an overview. Recent research also establishes that optimal control approaches is fast enough for it to be used for (receding horizon) feedback control in the near future [167].

The use of optimal control as a planning algorithm suffers from three issues. First, it relies on knowledge of the boundaries of the obstacles to guide the trajectory between them. Such a description is often difficult to obtain, especially if the robot moves through new environments, with objects it has not encountered before. Second, optimal control algorithms often invest significant computation time to go from a feasible, or good enough, solution to the optimal solution. In some cases this is necessary, such as in the extreme scenarios studied in the self-stability and dissipative-action related parts of this thesis. These scenarios were only feasibly by (nearly) optimal movement. For everyday scenarios, performing a task while not hitting obstacles nor making overly wasteful or slow movements, is already satisfactory. The computation time can be reduced, by simply stopping the search early, but this requires feasible intermediate solutions, which is not always possible. The feasibility of solutions also relates to the third issue with using optimal control for planning: the methods search locally, meaning they can get trapped in local optima or fail to find feasible solutions.

A different approach to planning, so-called sampling-based planning solves these two issues [47]. Sampling based planners have weaker convergence guarantees than their non sampling-based counterparts, and often do not provide optimal

solutions. However, they can be much faster in practice and do not require a geometric description of the obstacles. The idea behind sampling based planners is to try randomly selected sequences of inputs (actions), while keeping track of progress towards the goal state.

The most commonly used method of tracking this progress is building a graph of nodes in the planning space, that is in the configuration or state space. The nodes in the graph signify those positions that have been reached without hitting obstacles by trying the random actions. The difference between various algorithms is in selecting where and how to expand the graph, and to what extend it is tried to connect a new node to the other nodes in the tree. The three most used variations are called Probabilistic Roadmaps (PRM, [77]), and two variations of Monte Carlo Tree Search (MCTS, [22]) namely Upper Confidence Bound applied to Trees (UCT, [84]) and Rapidly exploring Random Trees (RRT, [92]).

The difference between PRM and MCTS is the graph structure the algorithms create. PRMs build a graph structure that allows loops, which allows reuse of the graph. However, in a changing environment reuse requires efficient recomputation of the feasibility of the motion segments, as in [187]. Such recomputation has not been established for planning in state space yet. As such, MCTS approaches, which creates simpler graphs (trees) for each query are a better fit for planning dynamic movement in changing environments

The most famous branch of MCTS-algorithms are the UCT algorithms, which are very generic, and therefore well suited to find high quality plans in domains about which little insight exists, hence this type of search is particularly known for use in games [29]. It has also been used for robotics [33], in situations where more specialized algorithms are not available. One such instance is in Chapter 6 of this thesis, where a similar algorithm is used.

For robotic trajectory planning however, RRTs are potentially preferable over UCTs, as they are more specialized for that purpose. In particular, they are well suited to exploring the solution space for motion planning problems. For its rapid exploration, RRT algorithms rely on a (pseudo)metric which approximates the distance between two nodes. The exploration is further enhanced when the RRT has access to a local planner, i.e., a function that can find the correct steering input for short motion segments.

The two requirements to make RRT work well are hard to satisfy in trajectory planning. The distance function associated with an optimal control algorithm is a sub-Finslerian metric [99], so is not continuous and not symmetrical. Experiments have shown that using the Euclidean distance to approximate this distance leads

to poor results [154]. Furthermore, local planning algorithms are often based on iterative schemes, and therefore not fast enough to use in this setting.

To alleviate the problems with the distance functions, many heuristic approximations have been devised. A succesful approach is to use the use reachability to find out whether Euclidean distance is sensible [154]. When it is sensible, the Euclidean distance is used, otherwise the distance is assumed infinite. While these approaches are useful, they may not be accurate enough for planning more challenging dynamic motions.

To obtain a more accurate approximation of the distance function, recently machine learning has been used [17, 122]. The machine learning generalizes from a database of trajectories, with associated costs. In principle, the steering function can also be approximated this way, although this has not yet been shown in literature.

The main challenge with this learning based approach to planning is to generate the trajectories. The function approximation needs many samples, so most optimal control schemes take a long time to compute them. There are two main approaches to optimal feedforward control: direct and indirect. The most popular approach first discretizes the functions that are optimized, in order to use more standard numerical optimization techniques to find a solution. This so called direct approach is more numerically stable, and is therefore preferred in most current literature, including in the research on learning RRTs [142]. Examples of approaches and applications using this direct approach are algorithms such as single shooting, multiple shooting, various collocation methods and the ILQG/ILQR approach.

The alternative, so-called indirect, approach optimizes analytically, and subsequently discretizes the resulting differential equations [117]. The indirect approach originated from the classic Euler-Lagrange approach, and is now applied as the maximum principle of Pontryagin [135]. This approach is numerically less stable, and therefore mostly used in a theoretical setting. However, it has some properties that might make it well suited for use in RRTs.

Therefore, the part of this thesis that aims to improve adaptability investigates the following hypothesis: *indirect optimal control improves the performance of learning-type path planners.*

1.5 | Research questions and thesis outline

This section summarizes the research questions posed in the previous sections, and explains what the role of each chapter is in answering them.

- Is it possible to make a self-stabilizing robot arm perform a pick-and-place task?

  Chapter 2 The goal of this chapter is to show that feedforward control by itself can make robot arms exhibit a stable limit cycle. To that end, I develop a feedforward controller for a two degree of freedom robot arm such that it performs a stable motion without the use of feedback controllers or potential energy shaping.

  Chapter 3 The goal of this chapter is to make a robustly stable robot arm, such that a robot with limited sensing can learn to perform a self-stabilizing motion, despite both model uncertainty and disturbances. To that end, I develop a controller and a learning scheme that robustly stabilizes a two degree of freedom robotic arm performing a reaching task without feedback, after an initial learning phase.

  Chapter 4 The goal of this chapter is to extend the work of the previous chapter, such that the robot arm can perform a pick-and-place task.

- Is optimizing the feedforward controller for self-stability a useful approach to improving the accuracy of robot arm motions?

  Chapter 5 In this chapter, a PID-controlled robot arm either maximizes or minimizes its accuracy by changing the movement it uses. The accuracy of those motions is studied for different feedback gains.

- Can robot arms perform reaching motions with only dissipative actions?

  Chapter 6 In this chapter, a robot arm whose actuators cannot inject energy in the system is controlled to perform a reaching motion.

- Does indirect optimal control improve the performance of learning-type path planners?

  Chapter 7 In this chapter, the RRT CoLearn algorithm, which combines supervised learning with indirect optimal control, and shows a tenfold decrease in planning time compared to earlier learning RRT algorithms.

  Chapter 8 In this chapter, two further arguments for the use of indirect optimal control and learning based RRT are provided. First, it provides a proof of probabilistic completeness of learning based RRT. Second, it shows a method to sample optimal trajectories using indirect optimal control that is valid for an important class of robotic systems.

The final chapter reflects upon the first eight by providing discussion and the main conclusions.

2

Open Loop Stable Control in Repetitive Manipulation Tasks

Michiel Plooij*, Wouter Wolfslag* and Martijn Wisse
* These authors contributed equally to this chapter,
IEEE International Conference on Robotics and Automation, 2014.

Abstract

Most conventional robotic arms depend on sensory feedback to perform their tasks. When feedback is inaccurate, slow or otherwise unreliable, robots should behave more like humans: rely on feedforward instead. This chapter presents an approach to perform repetitive tasks with robotic arms, without the need for feedback (i.e. the control is open loop). The cyclic motions of the repetitive tasks are analyzed using an approach similar to limit cycle theory. We optimize open loop control signals that result in open loop stable motions. This approach to manipulator control was implemented on a two DOF arm in the horizontal plane with a spring on the first DOF, of which we show simulation and hardware results. The results show that both in simulation and in hardware experiments, it is possible to create open loop stable cycles. However, the two resulting cycles are different due to model inaccuracies. We also show simulation and hardware results for an inverted pendulum, of which we have a more accurate model. These results show stable cycles that are the same in simulation and hardware experiments.

2.1 | Introduction

The vast majority of robotic manipulators require sensory feedback in order to perform their tasks. Humans, on the other hand, use both feedback and feedforward (or open loop) control when controlling their body [42]. Using feedforward allows humans to control their body despite having large time delays (typically 150 ms [36, 170]). Although robots have faster feedback loops, feedforward still has advantages. First, feedforward can anticipate on future states of the system, second, is can offer cheaper control when the cost is critical, and third, it is suitable for systems with slow and imprecise feedback, such as camera based feedback.

In a previous study, we showed the remarkable result that sensitivity to some modeling inaccuracies can be eliminated by choosing the right feedforward controller [133]. This result was similar to observations in humans, who minimize the influence of uncertainty on the final position of feedforward controlled movements [62]. Where the previous study only considered short motions (one second), this chapter takes those ideas a step further by considering long term stability of open loop controlled robotic arms (see Fig. 2.1), inspired by (human) stable walking motions.

One of the commonly cited disadvantages of open loop controllers is that they cannot directly compensate for perturbations, since those perturbations are not

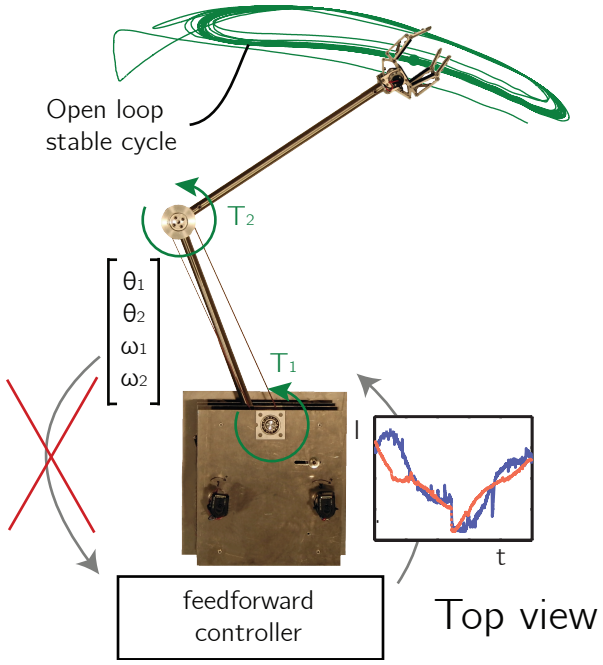


Figure 2.1: This figure shows the top view of the concept of open loop stable manipulation for our two DOF robot setup. Since there is no feedback available, the torque signal is a function of time only. Using numerical optimizations, we find torque signals that result in open loop stable cycles and allow the robot to perform repetitive tasks. Both the path displayed and the current signals are obtained from hardware experiments.

fed back into the controller. Since most tasks of robotic arms are repetitive, we propose to view them as cycles and consider the inherent stability of those cycles. If the trajectories we perform are stable, perturbations will simply decay over time, without the need for sensory feedback.

In literature, stability is defined in different ways, depending of the systems and scenarios studied. In this thesis, stability is taken as assymptotic stability with respect to a desired nominal trajectory, see [145] for the mathematical definition. In practice, assymptotic stability means that if the robot starts away from the nominal trajectory, it will converge to that trajectory, with the distance to that trajectory at all times bounded.

As such, obtaining assymptotic stability is a first step towards handling disturbances: it shows a single disturbance is corrected over time. Key in this study is that a desired nominal trajectory is stabilized. Many physical systems will converge to some motion given a feedforward input, for instance, a pendulum with

zero input will converge to its vertical hanging position. However, these motions will typically not accomplish the desired task, such as reaching a desired position at a set time.

For task performance, more properties than stability are important: such as the rate of convergence, and the bounds on the distance from the trajectory after disturbances. Furthermore, the sensitivity of task performance to errors in the trajectory, and sensitivity to uncertainty in the model are important factors in final performance. Due to the, previously stated, difficulty associated with open loop control and disturbance rejection, this chapter focusses on the first step: obtaining assymptotic stability.

Other researchers have already used the inherent stability of specific motions in order to create functional robots. Those robots can be split into two groups: robots with only limited state feedback and robots without any state feedback.

A well known example of robots with limited state feedback is given by Schaal and Atkeson [146], who studied open loop stable juggling with a robotic arm. In their case, open loop means that the state of the ball is not used as an input for the controller, but the arm itself is position controlled. Other examples include a timed position controlled swing leg retraction to stabilize running [152] and rope turning without measuring the state of the rope [80].

The group that is more related to this study is the group of robots without any state feedback. The most striking result in this group was obtained by McGeer, who introduced the concept of passive dynamic walking [109].Those walkers do not have motors and thus they do not use any feedback control, while their walking motion is stable. The stable walking motions do not rely on the motion being stable at each point in time, rather they work due to the existence of stable cyclic motions, called limit cycles. Such cycles were later on used in so called limit cycle walkers in combination with feedback control [59, 66, 71]. The work most strongly related to this chapter is that by Mombaur et al. [113, 114]. They found stable open loop controllers for walking and running robots by optimizing the open loop controllers for both stability of the motion and energy consumption. These results on a variety of systems indicate that open loop stable control can be an effective approach for robotic arms as well.

However, the cited works all perform tasks in which impact plays a central role. Under the right conditions, impact can have a large stabilizing effect. For the reaching task the manipulator studied here has to perform, no such impact is available. This gives rise to the question: can a feedforward controller by itself stabilize reaching motions? This chapter shows the answer is yes.

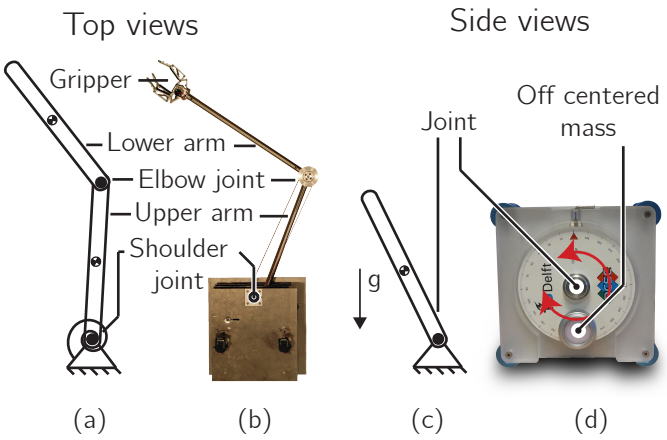


Figure 2.2: The four configurations we studied: (a) a simulation model of a two DOF robotic arm, (b) a two DOF robotic arm, (c) a simulation model of an inverted pendulum and (d) an inverted pendulum.

Therefore the goal of this chapter is to introduce a new approach in robot manipulator control: *open loop stable manipulation* and to show the first results using this approach. We show results both in simulation studies and in hardware experiments.

The remainder of this chapter is structured as follows. Section 2.2 explains the configurations we studied and the optimization method we used. Section 2.3 shows the results for the two DOF robotic arm, including a description of the simulation model, the simulation results and the hardware results. Section 2.4 shows the results for the inverted pendulum, including a description of the simulation model, the simulation results and the hardware results. Finally, the chapter ends with a discussion in Section 2.5 and a conclusion in Section 2.6, where we will conclude that it is possible to perform manipulation tasks with an open loop controller by performing open loop stable cycles.

2.2 | Methods

We studied open loop stable task execution of robotic arms by optimizing the open loop controller such that the task is performed by a stable cyclic motion. In this section, we discuss the configurations we studied and the optimization method.

2.2.1 Configurations

We studied four configurations, which are also shown in Fig. 2.2:

(a) **A simulation model of a two DOF robotic arm**. On this model we optimized the open loop controller such that the arm makes stable cyclic motions while performing the specified task.

(b) **A prototype of a two DOF robotic arm**. We implemented the controller obtained in (a) on a two DOF robotic arm to test how the controller performs on a real system. The results will show that the open loop controller generates stable cycles on the robot, but does not converge to the same trajectory as in (a). In Section 2.5.2, we will argue that this is caused by a bending of the second DOF due to gravity.

(c) **A simulation model of an inverted pendulum**. We used the same techniques as used in (a) to obtain open loop stable motions of an inverted pendulum. We show that on this system, the results have an intuitive explanation.

(d) **A prototype of an inverted pendulum**. We implemented the controller obtained in (c) on an inverted pendulum to test how the controller performs on a real system for which an accurate model is known.

2.2.2 Optimization method

The stability of limit cycles can be assessed with a number of different measures, such as Lyapunov stability [162] or contraction analysis [97, 103]. We used the classic notion of Poincaré maps [162] to find the stability of the open loop controlled motions, which we will now explain.

Consider a non-linear system described by the following differential equation:

$$\dot{x} = f(x, u(t)) \tag{2.1}$$

Since we use open loop control, we can consider the time as an extra state. Using transverse coordinates [9, 102, 153] with the time as phase variable in this appended state space is the same as using error dynamics in the original state space. This means that we can set a Poincaré section at $t = t_f$. We calculated the error dynamics along the trajectory $x^*(t)$ that results from the input $u^*(t)$. Since the controller is open loop, the error dynamics are simply given as the difference between the current state and the trajectory state, both at the same time:

$$\dot{x}^* = f(x^*, u^*(t)) \tag{2.2}$$

$$\delta x = x - x^* \tag{2.3}$$

$$\dot{\delta x} = f(x, u^*(t)) - f(x^*, u^*(t)) \tag{2.4}$$

Linearizing the dynamics along the trajectory results in:

$$\dot{\delta x} = \frac{\partial f(x^*, u^*(t))}{\partial x}\delta x \tag{2.5}$$

$$= A^*(t)\delta x \tag{2.6}$$

Where A^* is the linearized system matrix, and δx the error-state. We constrained the motions to be cyclic with period t_f (see below), which results in a cyclic $A^*(t)$ with the same period. Since system (2.6) is linear, we can take the state transition matrix Ψ from $t = t_0 = 0$ to $t = t_f$. Because we know $A^*(t)$ analytically, we can find Ψ by numerically computing the solution to the following initial value problem:

$$\dot{\Psi} = A^*(t)\Psi,\ \Psi(0) = I \tag{2.7}$$

Similar to the monodromy matrix in Poincaré map analysis of limit cycles, the motion is stable if the eigenvalues of Ψ have an absolute value smaller than one:

$$|\lambda\left(\Psi(t_f)\right)|_{\max} < 1 \tag{2.8}$$

To find a trajectory, we now use an optimal control approach similar to [113]. We used the above condition as a constraint rather than to minimize the left hand side of it, because of two reasons. First, the above condition specifies the convergence rate of the limit cycle, and does not specify other stability related factors, such as basin of attraction [67] and robustness against model uncertainties. Second, in practice other performance issues are also of concern, such as energy consumption and speed. We chose to use the integral of the squared input as objective, resulting in the following optimization:

$$\underset{u(t)}{\text{minimize}} \quad \int_{t_0}^{t_f} u(t)^2 dt \tag{2.9}$$

$$\begin{aligned} \text{subject to} |\lambda\left(\Psi(t_f)\right)|_{\max} &< 0.9 \\ |u(t)| &< u_{\max} \\ x(0) &= x_{\text{pick}} \\ x(t_1) &= x_{\text{place}} \\ x(t_f) &= x_{\text{pick}} = x(0) \end{aligned} \tag{2.10}$$

Where $u(t)$ is the input, $u_{\max}$ is the maximum input and x_{pick} and x_{place} are the pick and place states (see Section 2.3.2). The constraint that the final state is equal to the initial state causes the resultant motion to be cyclic.

In case of a setup with multiple motors, we used the integral of the sum of squared inputs as cost function. We used this cost function because it is often used in

other control applications. Furthermore, the resulting controllers are relatively smooth, whereas a time-optimal controller would be bang-bang. Such a bang-bang controller is undesirable in robot experiments, because it is more likely to be affected by unmodeled effects such as backlash. The optimization is performed using the optimal control package GPOPS [141] in Matlab.

2.3 | Two DOF manipulator

We implemented open loop stable manipulation on a SCARA type arm with two DOFs: two revolute joints moving two links in the horizontal plane (see Fig. 2.2b and 2.3). Since there is no spring present on the second joint of the arm, an open loop controlled motion must depend on dynamic effects for stabilization. Specifically, the zero input does not lead to a stable result. Since the dynamic effects are highly non-linear, it is not obvious whether stable motions exist. This section starts with a system description, including a description of the robotic arm and the simulation model. Next, we explain the task the arm has to perform, followed by simulation results and hardware results.

2.3.1 System description

Fig. 2.1 shows a picture of the two DOF robotic arm [131]. The DOFs are created by two 18x1.5mm stainless steel tubes, connected with two revolute joints, with a spring on the first joint. A mass of 1 kg is connected to the end of the second tube, which represents the weight of a gripper with product. The motors are placed on a housing and AT3-gen III 16mm timing belts transfer the torques within the housing. The joints are actuated by Maxon 60W RE30 motors with gearbox ratios of respectively 18:1 and 1:66. The timing belts provide an additional transfer ratio of 3:1 on the first joint and 5:3 on the second joint.

We used the TMT method [96] to obtain the equations of motion of the simulation model of this arm, which are too long to include in this chapter. The model includes 19 parameters, which are listed in Table 2.1. We included three types of frictional losses: Coulomb friction, viscous friction and torque dependent gearbox friction. Torque dependent gearbox friction is less commonly used than the other two, however, from the parameter values obtained through a system identification of the motor it is clear that this type of friction is not negligible (see Table 2.1). The way we implemented it is similar to the force dependent friction term in [46]. The friction in a joint is equal to:

$$T_f = -\mu_v \cdot \omega - \text{sign}(\omega) \cdot (\mu_c + \mu_t \cdot |T|) \tag{2.11}$$

for $\omega \neq 0$

$$T_f = -\min(\mu_c + \mu_t \cdot |T|; |T|) \cdot \text{sign}(T) \tag{2.12}$$

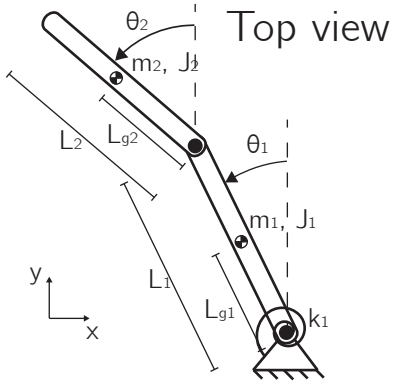


Figure 2.3: Top view of the two DOF system with a linear spring on the first joint. The second joint is actuated through a parallel mechanism (not shown in this figure), such that the angle of the second arm is an absolute angle. The friction acts on the absolute angles of the joints.

Table 2.1: The model parameters of the two DOF arm.

Parameter	Symbol	Value	Unit
Coulomb friction	μ_{c1}, μ_{c2}	0.0481, 0.0218	Nm
Viscous friction	μ_{v1}, μ_{v2}	0.03, 0.03	Nms/rad
Torque dependent friction	μ_{tf1}, μ_{tf2}	21.87, 31.91	%
Inertia	J_1, J_2	0.0233, 0.0871	kgm^2
Mass	m_1, m_2	0.809, 1.599	kg
Length	l_1, l_2	0.410, 0.450	m
Position of COM	l_{g1}, l_{g2}	0.070, 0.325	m
Motor constant	k_{t1}, k_{t2}	26.7, 28.1	mNm/A
Gearbox ratio	n_1, n_2	1:54, 1:110	rad/rad
Spring stiffness	k_1	1.6	Nm/rad

for $\omega = 0$.

ω is the velocity of the joint, T is the torque exerted by the motor on the joint, and μ_v, μ_c and μ_t are the coefficients of viscous, Coulomb and torque dependent friction respectively.

The simulation model includes a DC motor. The torque applied by the DC motor on the joint is equal to:

$$T = n \cdot k_t \cdot I \tag{2.13}$$

Where k_t is the motor constant, n is the gearbox ratio and I is the current through the motor. The current through the motor is constrained to 1 A (see eq. 2.10).

Coulomb friction and torque dependent friction introduce discontinuities in the equations of motion, which are difficult for GPOPS to handle. Therefore, we optimized the open loop controller on a model with only viscous friction, and added a torque afterwards to compensate for the Coulomb friction and torque dependent friction. Such a compensation can only be done when it does not effect the stability of the cycle, so when A^* is independent of both the compensated friction and the input. Both the Coulomb friction and the torque dependent friction depend on the state through a sign function, which is a piecewise constant function. Since A^* is the result of linearizing along the state, A^* does not depend on those friction terms (neglecting the always stabilizing effect of the discontinuity in the sign function). To make A^* independent of the input, we consider momenta instead of velocities. The equations of motion then become:

$$\dot{x} = f(x) + Bu(t) \tag{2.14}$$

With x, the state consisting of positions and momenta, and B a constant matrix. Following the definition in eq. (2.6), we see that A^* is therefore independent of u. Such a transformation is possible for many mechanical systems. Although such a transformation is not necessary, our specific optimization was faster with the transformation. For easier interpretation, we will show the velocities in the results.

2.3.2 Task

The robotic arm has to perform a pick-and-place task. The important task parameters are the pick state, the place state and the time per stroke. We show the results of the optimization for a motion which starts at $t = 0$ at the pick state x_{pick}, goes to the place state x_{place} at t_1 and then returns to the pick state at t_f with

$$x_{\text{pick}} = \begin{bmatrix} -0.7 \text{ rad} \\ -0.85 \text{ rad} \\ 0 \text{ rad/s} \\ 0 \text{ rad/s} \end{bmatrix}; x_{\text{place}} = \begin{bmatrix} 0.7 \text{ rad} \\ -0.3 \text{ rad} \\ 0 \text{ rad/s} \\ 0 \text{ rad/s} \end{bmatrix} \tag{2.15}$$

Where x is the vector consisting of the positions of the first and second arm, and their respective angular velocities. t_1 and t_f are free parameters in the optimization, but bounded as follows:

$$0.1 \text{ s} \leq t_1 \leq 1.2 \text{ s} \tag{2.16}$$

$$0.1 \text{ s} \leq t_f - t_1 \leq 1.2 \text{ s} \tag{2.17}$$

The goal is to find a path that statisfies the task constraints (that also include a stability-enforcing constraint) according to eq. (2.10), and then minimize the

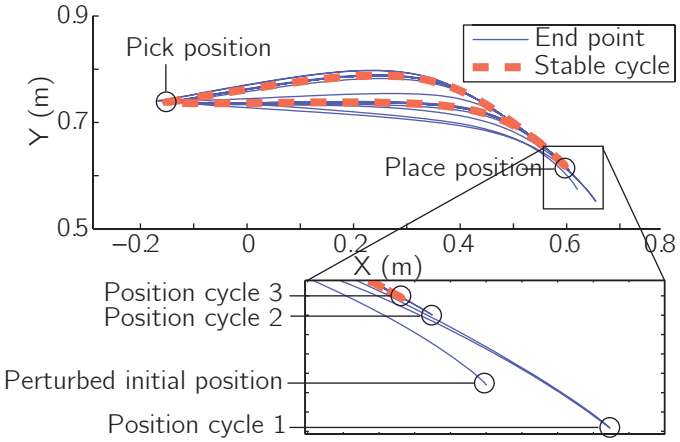


Figure 2.4: This figure shows the plot of the end point of the arm in simulation. The plot shows the stable cycle from simulation (thick dotted red line) and the motion of the robotic arm that starts from a perturbed position and converges to the stable cycle (thin blue line).

integral of the square of the current (see eq. (2.9)). Although we only show the result of one specific task, similar results were obtained using other task constraints. In none of these cases multi-starts or tuned initial conditions were needed, even though the optimization is non-convex.

2.3.3 Simulation results

Fig. 2.4 shows the position of the gripper in the workspace. The gripper does not start at the stable cycle, but converges to it. The sharp corners in the motion are the pick and place positions. Similar results were obtained for different pick and place positions.

Fig. 2.5 shows an example motion converging to the stable cycle in state space. The motion starts at a distance from the pick state $x_{\text{start}} = x_{\text{pick}} + [0.07; -0.15; 0; 0]$ and converges to the open loop stable cycle.

Fig. 2.6 shows the motion and current profile that result from the optimization. In Fig. 2.6a and 2.6b, we see that the motion starts and ends at the pick state, while at t $\approx$ 1.2 s, the arm is at the place state.

2.3.4 Hardware results

Fig. 2.7 shows the position of the gripper in the workspace. It clearly shows that the gripper does not start at the stable cycle, but converges to the cycle. Comparing the hardware results (Fig. 2.7) with the simulation results (Fig. 2.5) leads to the conclusion that although both the simulation and hardware results

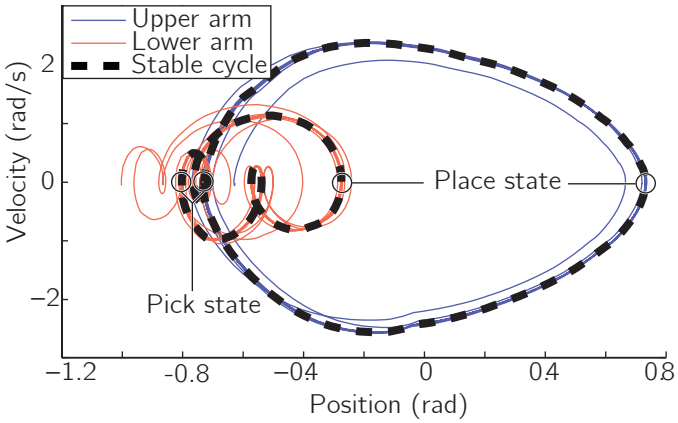


Figure 2.5: This figure shows the state space plot of the simulation data for the two DOF robotic arm. The plot shows the stable cycle (thick dotted red lines) and a motion that starts from a perturbed position and converges to the open loop stable cycle (thin lines).

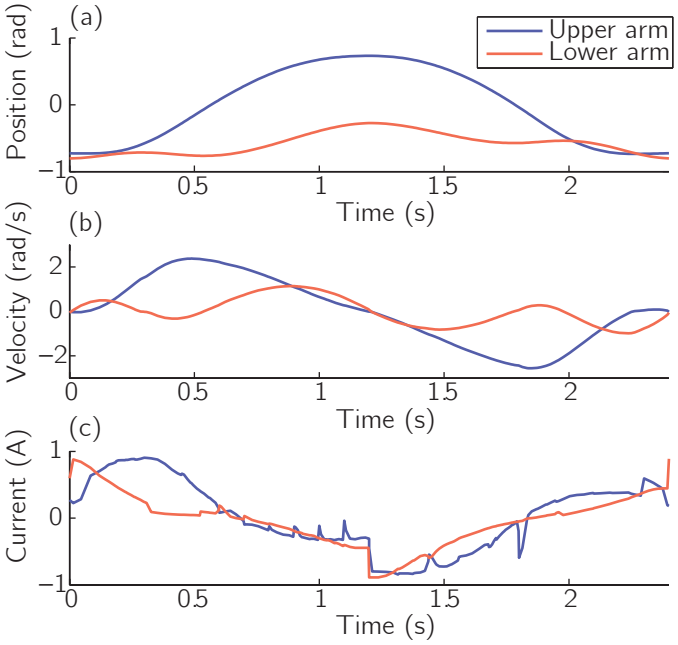


Figure 2.6: This figure shows the simulation data for the two DOF robotic arm as function of time. The figure shows the positions of the joints (a), the velocities of the joints (b) and the torques about the joints (c).

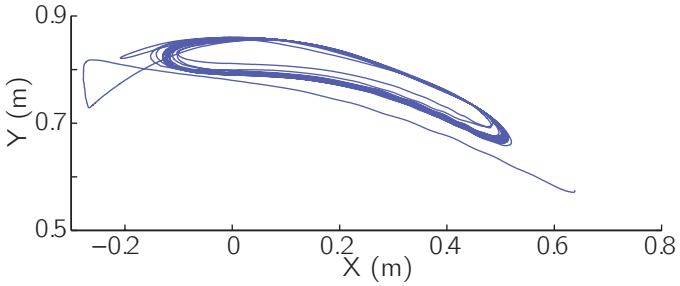


Figure 2.7: This figure shows the plot of the gripper in hardware experiments. The plot shows the motion of the robotic arm that starts from a perturbed position and converges to the stable cycle.

show convergence to a stable cycle, the stable cycles themselves are different. In Section 2.5.2 we will argue that the difference is probably caused by a bending of the second DOF due to gravity. In Section 2.4, we will show that for a simpler system (i.e. an inverted pendulum), our model is accurate enough to predict the exact cycle.

Fig. 2.8 shows the motion of the arm in state space. We see that the motion converges to a stable cycle in state space. When comparing Fig. 2.8 with Fig. 2.5, we see that range of positions of the upper arm is smaller in hardware experiments, and that the range of positions of the lower arm is larger in hardware experiments.

Fig. 2.9 shows the time series of a typical cycle of the robotic arm in hardware experiments. Again we see that the hardware results differ from the simulation results. In Fig. 2.9b, we notice that the velocity signals show a vibration at approximately 10 Hz. This vibration is caused by the elasticity of the timing belts between the motors (with encoders) and the joints.

The accompanying video shows a demonstration of the disturbance recovery of the two DOF arm. This video also indicates that the basin of attraction is large. In Section 2.5.2, we will show this basin of attraction in more detail.

2.4 | Inverted pendulum

In the previous section, we showed the results of a two DOF SCARA type arm. These results show that pick and place motions can be performed in an open loop stable manner. However, there was a difference between simulation and hardware results due to model inaccuracies. In this section we move to a simpler system, allowing us to both give an intuitive explanation for how the stabilization works, and indicate that with a more accurate model hardware experiments and

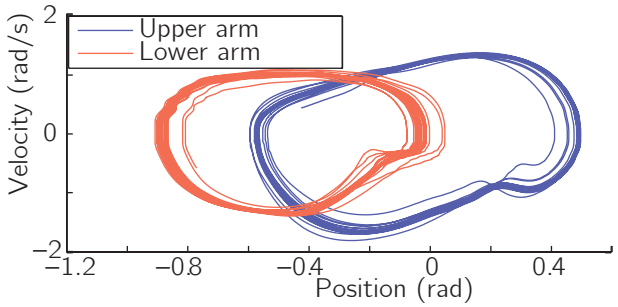


Figure 2.8: This figure shows the state space plot of the hardware experiments on the two DOF robotic arm. The plot shows the motion of the robotic arm that starts from a perturbed state and converges to the open loop stable cycle. In order to obtain a smooth graph, the velocity data is filtered with a fifth order Butterworth filter with the cutoff frequency at 10 Hz.

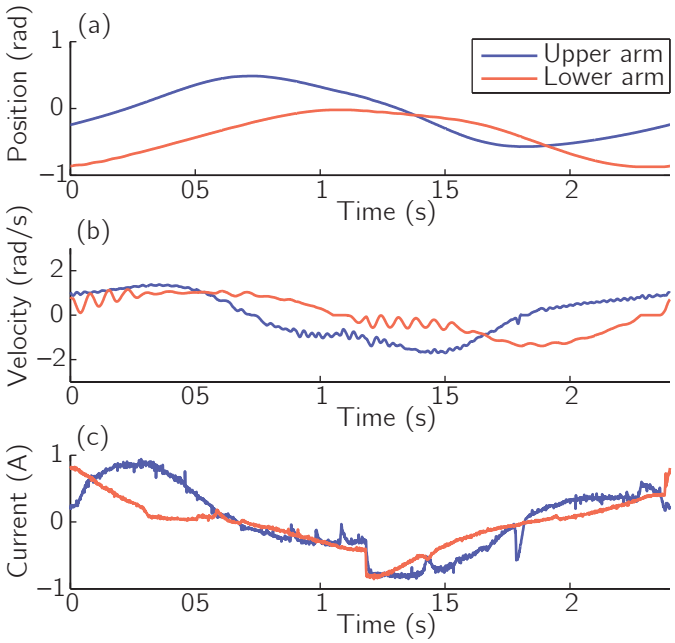


Figure 2.9: This figure shows the data of a typical motion of the hardware experiments on the two DOF robotic arm as function of time. The figure shows the positions of the joints (a), the velocities of the joints (b) and the current through the motors (c).

simulation can be made to match. For this purpose an inverted pendulum is used (see Fig. 2.10).

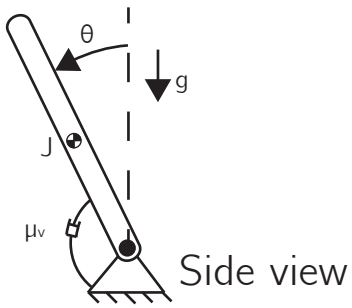


Figure 2.10: Side view of the one DOF system with linear viscous friction and gravity.

Table 2.2: The model parameters of the two DOF arm. The values are obtained through a system identification of the inverted pendulum.

Parameter	Symbol	Value	Unit
Gravitational term	c_1	112.9	s^{-2}
Motor parameters term	c_2	28.0	$V^{-1}s^{-2}$
Damping term	c_3	-1.8	s^{-1}

2.4.1 System description

Fig. 2.2d shows a picture of the inverted pendulum setup we used. The pendulum consists of a disk with an off-centered mass, which is connected to a DC motor without a gearbox in between. This direct drive actuation results in low friction, which makes the system easier to model. The DC motor is voltage controlled with a maximum voltage of 5 V (see eq. 2.10).

The differential equation for this system is

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \omega \\ c_1 \sin(\theta) + c_2 U + c_3 \omega \end{bmatrix} \tag{2.18}$$

Where, U is the input voltage, and c_1, c_2 and c_3 are the model parameters, which we identified through a system identification and are listed in Table 2.2. In this model, c_1 can be seen as the gravitational term, c_2 as the motor parameters term and c_3 as the damping term, which includes the back-emf term of the motor. All these terms have the inertia of the pendulum included.

We again used GPOPS to optimize an open loop controller such that we obtain an open loop stable motion for the task described below.

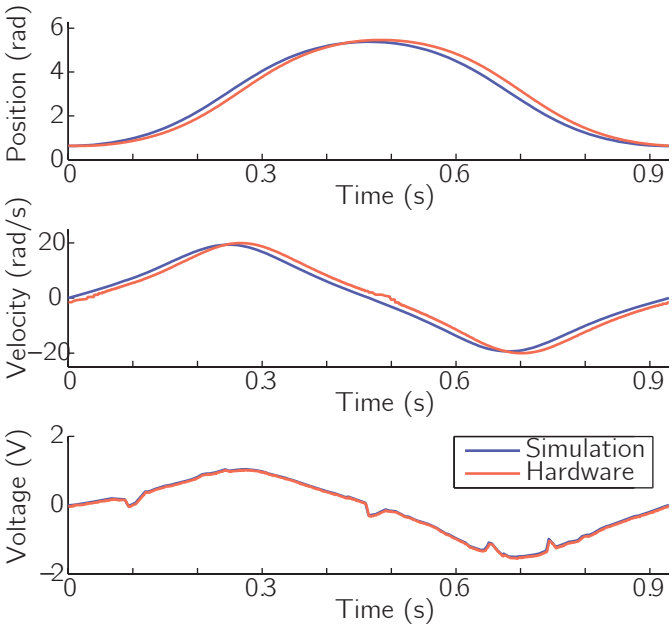


Figure 2.11: This figure shows the data for the single pendulum in the vertical plane as function of time. The figure shows the position of the joint (a), the velocity of the joint (b), and the torque about the joint (c) for both the simulation as the hardware experiments.

2.4.2 Task

The task we will look at is a motion with the initial and final position of the arm both at $^1/_5\pi$ (nearly upright position). Simply not moving will not result in stability, since by itself this is an unstable position. So at first, it seems impossible to find an open loop stable solution. However, this is possible when we allow the pendulum to swing to the region between $^1/_2\pi$ and $^3/_2\pi$ (around the lower equilibrium) during the motion. The goal is to find a path that is stable according to (2.8), and then minimize the integral of the square of the voltage (see eq. (2.9)). We limit the duration of the motion to 1.2 seconds.

2.4.3 Simulation results

Fig. 2.11 shows the result of the optimization, which can be explained intuitively. Since eq. (2.18) is linear in the velocity and the input, the A^*-matrix of the pendulum only depends on the position. That is, being in a state with θ between $-\pi/2$ and $\pi/2$ has a destabilizing effect, and being in a state with θ outside that range has a stabilizing effect. Loosely speaking, a stable motion requires that the stabilizing effects compensate the destabilizing ones. This means that the system

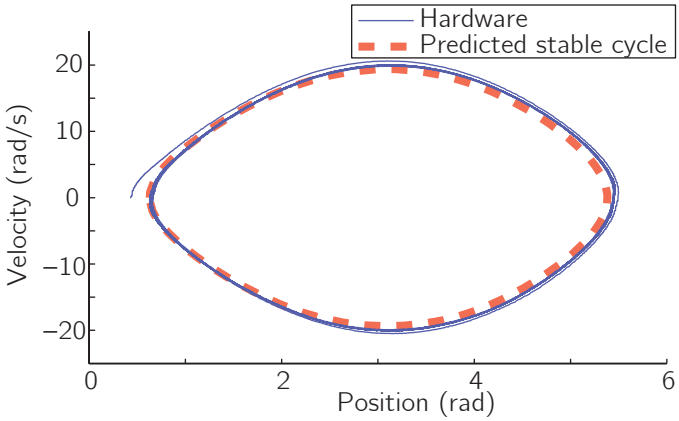


Figure 2.12: This figure shows the state space trajectory for the single pendulum in the vertical plane. The plot shows the stable cycle as determined in simulation (thick dotted line) and a motion on the hardware setup that starts from a perturbed position and converges to the stable cycle (thin line).

should spend enough time in sufficiently stabilizing positions to counter the time it spends in the destabilizing positions. In Fig. 2.11, we see that the pendulum moves directly from the destabilizing initial position to stabilizing positions, where it spends most of the time before moving back at the end of the motion. We use the condition in eq. (2.8) to evaluate if the stabilizing effects are compensating the destabilizing effects.

2.4.4 Hardware results

Fig. 2.12 shows the results of applying the input signal found in simulation on the hardware setup. It shows a motion being initialized in a perturbed position, after which it converges towards a cycle close to the one predicted by simulation. Fig. 2.11 shows that the motion over time after convergence is the same in simulation and hardware experiments. This shows that when an accurate model is available, the method we used finds a controller that performs the desired task on the real system. Extending the method to allow task performance even when accurate models are not readily available is an important next step in the research on open loop stable manipulation.

2.5 | Discussion

2.5.1 Model mismatch

For the two DOF robotic arm, the results from simulation (see Fig. 2.4, 2.5 and 2.6) are clearly different from those of the hardware experiments (see Fig. 2.7, 2.8 and 2.9). However, both simulation and hardware experiments show convergence to a stable cycle. These results show that the specific cycle the system converges to is sensitive to unmodeled behavior. For the inverted pendulum, we showed the results from simulation and hardware experiments are the same. This is due to the fact that the inverted pendulum is easier to model accurately.

There are three possible ways to reduce errors due to unmodeled behavior and thereby improve the control performance. First, the model of the system can be extended to include more of the unmodeled dynamics. Since modeling inaccuracies will always exist, a second approach would be to include the sensitivity to modeling inaccuracies in the optimization as in [133]. However, since sensitivity to modeling inaccuracies cannot always be reduced [133], we expect the best results from the third approach: tuning or learning open loop stable cycles online.

2.5.2 Basin of Attraction

In this chapter we focused on the calculation of open loop stability, which is a minimal requirement for making open loop manipulation work, but it gives little information about the rejection of realistic (i.e. finite) disturbances. Therefore, we simulated the arm using a grid of initial positions in order to see if they converge to the intended stable cycle. Fig. 2.13 shows this data, which is a slice of the 4D basin of attraction. This shows that the majority of initial positions (within the mechanical limits of the arm) result in convergence to the cycle. This result means that precise initial positioning of the arm is not required for converging to the intended cycle.

Interestingly, the basin of attraction analysis shows that all initial positions converge to the same cycle, although in some cycles, the second joint has rotated for exactly one or multiple revolutions. While implementing several feedforward controllers on the robotic arm, we found that most feedforward controllers (including the one shown in this chapter) result in two stable cycles: one with negative θ_2 and one with positive θ_2. We expect that this is caused by a bending of the arm due to gravity, which is larger around $\theta_2 = \pm\frac{\pi}{2}$ rad than around $\theta_2 = 0$ rad. We suspect that this difference between the model and the arm is the main cause for the mismatch of the cycles in simulation and in hardware experiments. In order to improve the prediction of the cycle by the model, a stiffer arm can be used or

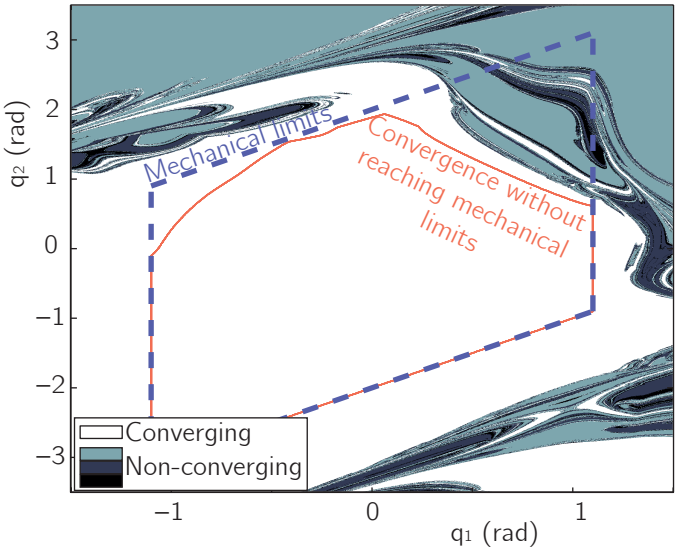


Figure 2.13: This figure shows the basin of attraction of the open loop stable cycle for different initial positions. The white region depicts all initial positions that result in convergence to the cycle. The blue dotted line depicts the mechanical limits of the robotic arm. All initial positions within the red line result in convergence while staying within the mechanical limits. The non-converging initial positions are divided into three groups. First, light blue depicts convergence to the intended cycle with an error in θ_2 of 2π rad; second, dark blue depicts an error in θ_2 of 4π rad and third, black depicts an error in θ_2 of 6π rad or more. So all initial positions converge to the same cycle, although in some cycles, the second joint has rotated for exactly one or multiple revolutions.

this bending can be modeled. A second cause of the model mismatch could be the friction, which is hard to model in general.

2.5.3 Implications

The idea of using open loop stability of periodic motions to analyze repetitive manipulation tasks is a new approach in robot manipulator control. It allows for robotic arms to be controlled when feedback is too slow (e.g. using camera feedback or control over great distance), too imprecise (e.g. due to cheap, noisy sensors [91]) or even impossible (e.g. in micro-scale applications or due to radiation), or when the input is limited and planning is required.

There is no fundamental reason why stable cycles would not exist in robotic arms with more DOFs, or for more complex tasks (e.g. obstacles or interactions with the environment). However, we expect that it will be more difficult to find such cycles and maybe impossible to find cycles that include the pick- and place-positions. Finding trajectories can be made easier by tuning the dynamics of the system,

such that all trajectories are stable, i.e. the system is contractive. Such tuning could be done by adding springs on all joints or changing the mass distribution.

2.5.4 Applicability

The concept of open loop stable manipulation as presented in this chapter is not fully applicable yet, since it consists of finding one stable cycle. In practice, tasks consist of moving between multiple positions in a certain (not necessarily predefined) order. Since the computations are too complex to perform online, we see two approaches to make open loop stable manipulation fully applicable in the future.

The first approach is to move in a stable periodic motion which covers most of the task space. Whenever necessary, the controller can make open loop controlled deviations from this cycle to the specified positions after which the manipulator returns to the main cycle.

The second approach is to divide the space into a grid and build a library of feedforward controllers for moving between the grid positions. The controller can then create various stable cycles by combining multiple trajectories from this library into a stable cycle that tracks the specified positions.

2.6 | Conclusions

In this chapter we introduced an approach to control robotic arms, without the need for state feedback. We conclude that this approach is promising for robotic arms that perform repetitive position tasks without feedback. Small disturbances on the state of the system will decay over time and the robotic arm will asymptotically return to its original trajectory. Both simulation and hardware experiments show convergence to an open loop stable cycle, but the cycles they converge to are not the same, probably caused by a bending of the second DOF due to gravity. We expect that this problem can be solved by online learning of the stable cycles.

Acknowledgement

This work is part of the research programme STW, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO).

3

Learning robustly stable open-loop motions for robotic manipulation

Wouter Wolfslag*, Michiel Plooij*, Robert Babuška and Martijn Wisse
* These authors contributed equally to this chapter,
Robotics and Autonomous Systems, Vol. 66, 2015.

Abstract

Robotic arms have been shown to be able to perform cyclic tasks with an open-loop stable controller. However, model errors make it hard to predict in simulation what cycle the real arm will perform. This makes it difficult to accurately perform pick and place tasks using an open-loop stable controller. This chapter presents an approach to make open-loop controllers follow the desired cycles more accurately. First, we check if the desired cycle is robustly open-loop stable, meaning that it is stable even when the model is not accurate. A novel robustness test using linear matrix inequalities is introduced for this purpose. Second, using repetitive control we learn the open loop controller that tracks the desired cycle. Hardware experiments show that using this method, the accuracy of the task execution is improved to a precision of 2.5cm, which suffices for many pick and place tasks.

3.1 | Introduction

This research aims at future applications where sensing and feedback are undesirable due to costs or weight; or difficult due to small scale, radiation in the environment or frequent sensor faults. A recent example of such an application is the control of a swarm of nano-scale medical robots [12]. These applications inspire us to investigate an extreme case of feedback limitations: solely open-loop control on robotic arms. Control without any feedback can only be effective if two key problems are addressed: disturbances (e.g. noise and perturbations) and model inaccuracies.

The first problem, handling disturbances on an open-loop controlled robot, has mainly been addressed by creating open-loop stable cycles. The best known examples of this are passive dynamic walkers, as introduced by McGeer in 1990 [109]. Since those walkers do not have any actuators, there is no computer feedback control. The walking cycle of those walkers is stable, which means that small perturbations will decay over time. Such stable cyclic motions are called limit cycles. Limit cycle theory was later used to perform stable walking motions with active walkers, of which the closest related work is that by Mombaur et al. [113, 114]. They optimized open-loop controllers for both stability and energy consumption and performed stable walking and running motions with those robots. Open-loop stable motions have also been used before to perform tasks with robotic arms. In 1993, Schaal and Atkeson showed open loop stable juggling with a robotic arm [146]. Even though their controller had no information about the position of the ball, they showed that any perturbation in this position decays over time, as long

as a specific path of the robotic arm itself can be tracked. In a recent study, we showed that it is possible to perform repetitive tasks on a robotic arm with solely an open-loop current controller [132].

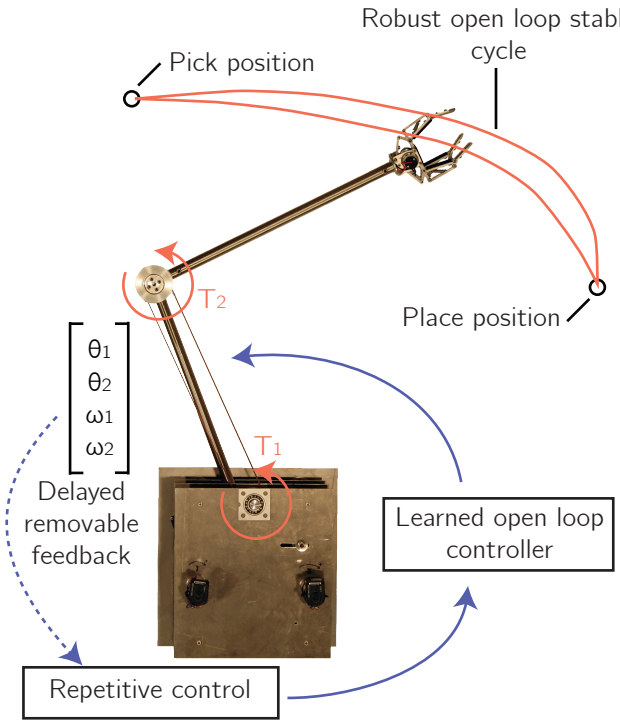


Figure 3.1: This figure shows the top view of the concept of robust open-loop stable manipulation. We first optimize a cycle that stands still at the pick and place positions for open-loop stability. Next, we check the cycle for robustness to model uncertainty. Then, using repetitive control on the robotic arm, we learn an open-loop controller that tracks the cycle. After the learning, the open-loop controller performs the task without any feedback.

The second key problem with feedforward control (i.e. model inaccuracies) prevents the approach in [132] to be fully applicable: it causes a difference between the motion as planned in simulation and as performed in hardware experiments. Handling model inaccuracies on open-loop controlled robots has recently become the subject of research. Singhose, Seering and Singer [155, 156] researched vibration reducing input shaping of open-loop controllers while being robust to uncertainty in the natural frequency and damping of the system. Becker and Bretl [11] researched the effect of an inaccurate wheel diameter of unicycles on the performance of their open-loop velocity controller. In their case, open-loop control means that the position of the unicycle is not used as an input for the controller, but the

velocity of the wheels is. In a previous paper, we showed that on a robotic arm, different open-loop current controllers have different sensitivities to model inaccuracies [133]. We found open-loop current controllers of which the end position of the motion is independent of the friction parameters. However, the motions that handle the model inaccuracy problem of feedforward control, the stability problem still exists, i.e., disturbances acting on these motions will grow over time.

Since these two problems of disturbances and model inaccuracies in open-loop control have only been addressed separately, no applicable purely open-loop control scheme has been devised. This chapter shows that repetitive tasks can be performed stably by robotic arms with an open-loop voltage controller, even when an accurate model is not available.

In order to achieve this goal, the problem is split into two phases (see Fig. 3.1). In the first phase the robustness of the system is analyzed with a novel method based on linear matrix inequalities (LMI)[150]. In the second phase repetitive control (RC)[98] is used to learn the exact control input, such that the desired positions are reached accurately. During this learning phase, very slow feedback is allowed, this feedback can be removed after the learning has been completed.

The rest of this chapter is structured as follows. Section 3.2 shows why the problem can be split into two phases, and explains the robustness analysis method and the repetitive control scheme. Next, Section 3.3 shows the experimental setup we used to test our approach. Then, Section 3.4 shows the results of both the numerical and the hardware experiments. Finally, the chapter ends with a discussion in Section 3.5 and a conclusion in Section 3.6.

3.2 | Methods

In this section we explain our methods. First, in Section 3.2.1 we discuss the basic concept of the stability analysis. Second, Section 3.2.2 explains our approach to perform robustly stable open-loop cycles. Then we will describe the two steps of this approach separately: a robust stability analysis (Section 3.2.3) and learning of an open-loop controller (Section 3.2.4).

3.2.1 Open-loop stable manipulation

A system described by the differential equation $\dot{x} = f(x, u)$ can be linearized along a trajectory x^* caused by input $u^*(t)$:

$$\frac{d\bar{x}}{dt} = A^*(t)\bar{x} \tag{3.1}$$

with

$$A^*(t) = \left.\frac{\partial f}{\partial x}\right|_{x^*(t),u^*(t)} \tag{3.2}$$

where $\bar{x}(t) = x(t) - x(t)^*$ is the state error. For the ease of notation, the time dependency of variables is occasionally dropped if it is unambiguous to do so. For example $A^*(t)$ will be written as A^*.

If both trajectory and input are cyclic with period t_f, stability can be assessed by by discretizing the system using a time step t_f. To be able to draw upon the research in stability of limit cycles, note that such discretization is the same as a Poincaré map of the system with the time appended to the state vector. The Poincaré section is then taken as $t = t_f$, and the time is reset to 0 after crossing this section. Previously (notably in [113] and [132]), verifying stability was done using the eigenvalues of the linearized discrete system. But that approach does not allow incorporating model uncertainty in the stability analysis.

To obtain a method that does allow uncertain models, we use a quadratic Lyapunov function, $J = \bar{x}^T M(t)\bar{x}$, with positive definite $M(t)$. The idea is that for a stable system, an $M(t)$ can be found such that the norm J is always decreasing over time. For cyclic systems this means the following two constraints should be satisfied (cf. [171]):

$$M(t)A(t)^* + \dot{M}(t) + A(t)^{*^T}M(t) \prec 0, \quad \forall t \in [t_0, t_f] \qquad \text{C1}$$

$$M(t_f) - M(t_0) \succ 0 \qquad \text{C2}$$

where $\prec$ and $\succ$ are used to indicate negative/positive definiteness respectively, $\dot{M}$ is the time derivative of M and the subscripts 0 and f denote initial/final time. The first of these constraints ensures that the Lyapunov function is decreasing at each time instant. The second constraint makes sure that it becomes stricter after each cycle, i.e., that having the same error ($\bar{x}$) as a cycle before means that the Lyapunov function has increased. Note that only one of the two inequalities needs to be strict in order for stability to hold.

When there are model inaccuracies, two changes occur that make the above conditions invalid. Firstly, $A^*(x^*(t))$ is no longer accurate when in state $x^*(t)$. Secondly, when using a fixed open-loop controller on an uncertain system, the trajectory is not fully predictable, so in general $x(t) \neq x^*(t)$, when using the input $u^*(t)$. In the next section we will outline our approach to solve these two issues.

3.2.2 Robust open-loop approach

To find motions that are open-loop stable even when the model is not accurately known, we will focus on input affine systems with constant input matrix, i.e., systems that are described by the following differential equation:

$$\dot{x}(t) = f(x(t)) + Bu(t) \tag{3.3}$$

where B is a constant matrix. The equations of motion of serial chain robots can be written in this form, by considering phase-space rather than state space (i.e. using momenta rather than velocities). Systems where the control input enters the system via a constant matrix have the advantage that the linearized error dynamics do not depend on u (cf. Eq. (3.2)):

$$A^*(t) = \left.\frac{\partial f}{\partial x}\right|_{x^*(t)} \tag{3.4}$$

Because the local stability of the motion only depends on the linearized error dynamics, the stability for motions of such a system only depends on the states of the motion, and not on the inputs used. This is the key insight that allows robust open-loop control, by splitting the problem into two stages:

1. Finding a trajectory through the phase-space that is stable, even if the (linearized) system dynamics differ from their nominal value by some uncertain amount.

2. Learning the inputs for that trajectory. When done online, this learning can take into account the uncertain dynamics.

With this two stage approach, an open-loop controller is found that accurately controls the robot in a way that is both accurate and stable. These two properties hold even when facing modeling errors, which is important for hardware implementation.

There are two important remarks to be made about this approach. Firstly, the robust trajectory found should be a valid trajectory for the physical system, and thus for all realizations of the uncertainty in the model. That means there should exist an input for which the trajectory given is a solution to the differential equation. In our case this means the inertia matrix has to be known accurately in order to translate momenta into the velocities that correspond to the time derivatives of the positions in the planned trajectories. Secondly, for the learning step some feedback is required, which means the robot is no longer purely open-loop

controlled. However, such feedback can be delayed and can be removed once the feedforward signal is known. This allows opportunities for external sensing, such as cameras based feedback during the learning phase.

In Section 3.2.3 the methods used to find a robust trajectory will be explained. Section 3.2.4 explains how Repetitive Control is used to learn the open-loop controller.

3.2.3 Finding robustly stable trajectories

The first step in our approach is to find trajectories that are stable. To do so, we use an optimization approach, further explained in Section 3.3. Then we test if the resulting trajectory is robustly stable. In this section we derive a novel robustness test, which is summarized in Algorithm 1.

For this section, we assume that we have a known trajectory for which we want to determine how robust it is. We model the uncertainty of the system through an integer number of uncertain parameters δ_j, that enter the linearization affinely using known state and time dependent matrices $\Delta_j(t)$, i.e.

$$A(t) = \left.\frac{\partial f}{\partial x}\right|_{x=x^*(t)} + \sum_j \Delta_j(t)\delta_j \tag{3.5}$$

where we will take $A^*(t) = \left.\frac{\partial f}{\partial x}\right|_{x^*(t)}$ as the certain part of the dynamics, and $\Delta(t) = \sum_j \Delta_j(t)\delta_j$ as the uncertain part. Here the Δ_j are known matrices describing how the uncertain parameters δ_j enter the system dynamics. As a result, we have a time varying uncertain system $\dot{\bar{x}} = A(t,\Delta)\bar{x} = (A^*(t) + \Delta(t))\bar{x}$.

From theory of Linear Matrix Inequalities [150, Prop. 5.3], we know that if each δ_j is constrained to some interval $\delta_j \in [\underline{\delta}_j, \bar{\delta}_j]$, then the constraint C1 holds for all $\Delta(t)$, if it holds for the $\Delta(t)$ created by the vertices of the hypercube of allowed δ_j. The $\Delta(t)$ values connected to these vertices will be called $\Delta_0(t)$, which is a finite set. The choice for $\Delta_0(t)$ depends on the expected model inaccuracies. The robust stability constraint now becomes:

$$MA^* + \dot{M} + A^{*T}M + \Delta_0^T M + M\Delta_0 \prec 0 \qquad \text{C3}$$

Note that this constraint is a sufficient, but not necessary condition for robust stability. Making M a function of Δ would reduce the conservativeness [150]. However, in implementation this would also result in much greater complexity and computation time, so we have elected not to do so.

Furthermore, constraint C3 determines whether the trajectory is robustly stable or not, but it does not give a continuous measure of robustness. To add this measure, we try to find the maximal constant ϵ_x, such that the following constraint holds:

$$MA^* + \dot{M} + A^{*T}M + \epsilon_x \Delta_0^T M + \epsilon_x M \Delta_0 \prec 0 \qquad \text{C4}$$

The value of ϵ_x is the robustness measure.

What is left is a way to find an $M(t)$ that satisfies the constraints. Before describing our approach, we will briefly discuss two approaches that involve optimizing a paramet rized $M(t)$, and are unsuitable in this case, but at first sight might seem applicable.

The first of these approaches is the most straightforward conceptually and has been used in literature [32, 101, 171]. The idea in that research is to parameterize $M(t)$, and then optimize these parameters. The literature referred to establishes that this parameter optimization can be cast as a convex problem by using Sum Of Squares programming. The disadvantage of this approach for our problem is that the optimization of $M(t)$ requires many (> 1000) decision variables and is therefore computationally expensive and sensitive to numerical errors.

The second approach is to parameterize $M(0)$ only, and rework equation C1 or C3 into a differential equation, which can be integrated to find $M(t)$. The basic example would be

$$\dot{M} = -\epsilon_s I - A^{*^T} M - M A^* \tag{3.6}$$

with ϵ_s a small positive constant. This allows to optimize $M(0)$, such that $M(t_f)$ found using integration satisfies equation C2. This approach of integrating $\dot{M}$ could potentially be extended to incorporate the uncertain dynamics into the differential equations, for instance by viewing M as an ellipsoid, and using a minimum volume ellipsoid covering algorithm. However, even then the method of searching for $M(0)$ and using forward integration is troublesome. The number of parameters is greatly reduced compared to approaches where $M(t)$ as a whole is approximated and optimized, but at the cost of losing convexity. This greatly increases the computation time and introduces the risk of finding local minima.

Since these two approaches that involve searching for $M(t)$ do not work, we propose a method which does not involve such optimization, but rather immediately computes a (suboptimal) $M(t)$. This method is based on a result from Floquet theory, which states that all time varying systems with periodic coefficients are reducible [53, Sec. XIV.3], i.e. can be transformed into a time invariant system. The idea is to find a Lyapunov function for the certain part of this time invariant

system and then transform this Lyapunov function back to the time variant system for the robustness check.

The transformation to the time invariant system is based on the state transition matrix of the nominal system $\bar{x}(t) = \Phi(t)\bar{x}(0)$, which can be found by integrating the following initial value problem:

$$\dot{\Phi}(t) = A^*(t)\Phi(t), \quad \Phi(0) = I \tag{3.7}$$

Now if we define $L(t) = \Phi(t)\Phi(t_f)^{-\frac{t}{t_f}}$ and the transformation $\bar{x} = L(t)y$, we get for the state transition matrix of y:

$$\dot{y} = \frac{1}{t_f}\ln(\Phi(t_f))y \tag{3.8}$$

which is a time invariant, but possibly complex-valued system [53, Sec. XIV.1]. For this system we can then find a Lyapunov function $y^T M_y y$ by solving for the positive definite matrix M_y using standard LMI techniques. In principle it would be possible to incorporate a worst-case transformed $\Delta(t)$ in that equation, but for simplicity we optimize to find a Lyapunov function with the smallest time derivative, i.e. maximize ϵ_y in:

$$\frac{1}{t_f}\ln(\Phi(t_f))^T M_y + M_y\frac{1}{t_f}\ln(\Phi(t_f)) + \epsilon_y I \prec 0 \tag{3.9}$$

The resulting M_y is then transformed back to $\bar{x}$ coordinates:

$$M(t) = L(t)^{-T} M_y L(t)^{-1} \tag{3.10}$$

Finally, the maximum ϵ_x that satisfies constraint C4 at all times is found using a numerical LMI solver. The time derivative of M that is needed for this step is readily determined analytically. We omit these expressions because they are too lengthy.

As Eq. (3.7) can only be solved numerically, a natural time sampling approach is used. That is, the constraint C4 is only tested at the sampling times that are used by the solver that integrates Eq (3.7). In this case, we used the MATLAB ode45 solver. This time sampling is inspired by [171], which discusses the correctness of such a sampling procedure for a similar robustness test.

In many cases one or more of the singular values of the state transition matrix $\Phi(t_f)$ are nearly zero [144]. This means that some errors are reduced to nearly zero after one cycle. This can happen for instance with a high damping constant, which could arise from using voltage control.

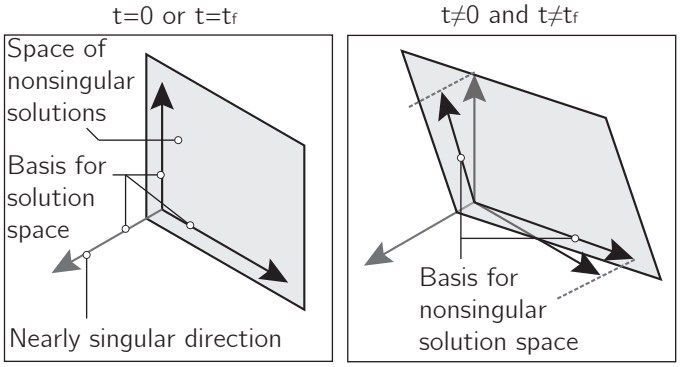


Figure 3.2: Explanation of the transformations required for model reduction. At time $t = 0$ or $t = t_f$, the singular value decomposition can be used to find a basis for the solutions space. In the left figure it can be seen that this basis has the advantage that the singular directions are clearly separated. The right figure shows that by using the same basis at other times, the non-singular dimensions are described using more basis vectors than are required to span the space. The solution, also indicated in the right figure is to use a projection, in our case by computing the column echelon form. This particular projection ensures the basis is periodic, which allows the robustness analysis to be performed.

Such a nearly singular state transition matrix makes the numerics of the above scheme ill-conditioned. Furthermore, the resulting LMI-test is very sensitive to small changes in the nearly singular directions. This numerical sensitivity is in contrast with the fact that in many cases, such near singularity is some intrinsic property of the system or trajectory, and is not influenced by some small changes in model parameters. In those cases the above scheme is sensitive to errors that are irrelevant. Therefore we disregard these near-singular directions by assuming the error in those directions is always 0.

Since the system is time varying, the relevant directions are also time varying. So what is needed is to find a time varying basis for the non-singular space of the error dynamics, see Fig. 3.2. The robustness test described previously requires a periodically time varying system, therefore the basis for the non-singular dimension should also be periodically time varying.

First, to disregard the near singular directions, use the singular value decomposition of the state transition matrix at time t_f: $\Phi(t_f) = U\Sigma V^T$. Define U_χ as the columns of U that correspond to the singular values that are not nearly 0. These columns are an orthonormal basis for the not-nearly-singular space of $\Phi(t_f)$. In particular, define $\tilde{x}(t)$ coordinates by the transformation $\bar{x} = U\tilde{x}$. Then

$$\tilde{\Phi}(t) = U^{-1}\Phi(t)[U_\chi, 0] \tag{3.11}$$

is the state transition matrix for $\tilde{x}$, where the errors in the singular dimensions are immediately set to zero (by the multiplication with $[U_\chi, 0]$). The columns of $U\tilde{\Phi}(t)$ now form a time varying basis for the non-singular space of the $\bar{x}$-dynamics. However, this basis is not periodic yet.

This periodicity can be enforced by using the following transformation, which defines $\hat{x}(t)$:

$$\bar{x}(t) = U\text{rcef}\big(\tilde{\Phi}(t)\big)\,\hat{x}(t) \tag{3.12}$$

where rcef signifies the reduced column echelon form. To see that it works, it is necessary to realize that after exactly one cycle, the columns of $\tilde{\Phi}$ span the same space. Furthermore, $\tilde{\Phi}(0)$ is simply the identity matrix with the last diagonal entries turned into 0. Therefore $\tilde{\Phi}(t_f)$ also spans this same space, and the reduced column echelon form then finds the correct basis: the identity matrix with the last entries turned into 0. As a result $\text{rcef}(\tilde{\Phi}(0)) = \text{rcef}(\tilde{\Phi}(t_f))$. This means that the transformation in Eq. 3.12 is periodic, and can therefore be used in the robustness test. The procedure of the robustness test, including this transformation is denoted in Algorithm 1, in which each step is described and the relevant equations for that step are then referred to.

This leaves only two remarks on the implementation of this transformation. First, the reduced column echelon form of a smoothly varying matrix is smoothly varying, meaning the time derivative of the transformation exists. To speed up computation we use finite differences to compute this derivative. Second, the transformation as defined above does not actually reduce the number of states in the computation, but rather sets the states to zero. By substituting U_χ for $[U_\chi, 0]$ in Eq. 3.11, the size of the state vector is reduced, which again speeds up computation.

Algorithm 1 Robustness test

Compute state transition matrix $\Phi(t_f)$, Eq. (3.7)
if nearlySingular($\Phi(t_f)$) **then**
    Reduce system dimension, Eqs. (3.11)-(3.12)
Make system time invariant by transformation, Eq. (3.8)
Find M_y, the Lyapunov function for that system, Eq. (3.9)
Initialize robustness measure, $\epsilon_x \leftarrow 1000$
for all t in time discretization **do**
    Find $M(t)$, Lyapunov function of first system, Eq. (3.10)
    Numerically find $\dot{M}(t)$ as required in constraint C4.
    $\bar{\epsilon} \leftarrow$ maximum ϵ_x that satisfies constraint C4 at time t
    $\epsilon_x \leftarrow \min(\bar{\epsilon}, \epsilon_x)$

3.2.4 Repetitive control

The stability of the trajectories computed by the optimization is independent of the input and finite modeling errors. The next step is to learn the input that tracks the trajectory on the robotic arm. When the trajectory is learned, the feedback is disconnected and the task is performed stably with the learned open-loop controller.

There are two (similar) learning algorithms which are commonly used for learning of repetitive motions: Iterative Learning Control (ILC) and Repetitive Control (RC) [98]. Both algorithms use the input and error from the previous iteration (cycle) to compute the input in the current iteration. The only difference between the two is that in ILC every iteration starts at the same state, where in RC every iteration starts at the final state of the previous iteration. We use RC, because it allows us to immediately start a new iteration at the end of every movement cycle, whereas ILC would require a reinitialization after every cycle to start the new iteration, which would take time and require fast and precise state feedback. The repetitive control runs on a real time target with a sampling frequency of 500 Hz. Therefore, we will use discrete time notation with time step k.

The repetitive control scheme we used is

$$u(k) = u(k-p) + \alpha(k) \cdot (\Delta u_P(k) + \Delta u_D(k)) \tag{3.13}$$

with

$$\Delta u_P(k) = \sum_{i=0}^{p-1} \phi(i) \cdot P \cdot e(k-p+i) \tag{3.14}$$

$$\Delta u_D(k) = \sum_{i=0}^{p-1} \sigma(i) \cdot D \cdot (e(k-p+i) - \bar{x}(k-2p+i)) \tag{3.15}$$

$$\bar{x}(k) = x^*(k) - x(k) \tag{3.16}$$

where $u(k)$ is the control input, p is the number of time steps in one iteration, $\bar{x}(k)$ is the error, x^* is the desired state and x is the actual state. $\alpha(k)$, $\phi(i)$, $\sigma(i)$, P and D are explained below.

The learning rate $\alpha(k)$ is a function of time and can vary between 0 and 1. A varying $\alpha(k)$ reduces the chance of converging to a sub-optimal solution.

The filter gains $\phi(i)$ and $\sigma(i)$, for $i = 1, \ldots, p$, determine how much the different errors in the previous iterations contribute to the change in the input signal. The filtering accounts for measurement noise and prevents oscillations in the RC. The sum of the elements of $\phi(i)$ and the sum of the elements of $\sigma(i)$ are both equal to 1, to obtain a (weighted) moving average filter.

The learning gains P and D are $N_u \times N_s$ matrices (with N_u the number of inputs and N_s the number of states). The P and D we used have the following structures:

$$P = \begin{bmatrix} P_{11} & 0 & P_{13} & 0 \\ 0 & P_{22} & 0 & P_{24} \end{bmatrix} \tag{3.17}$$

$$D = \begin{bmatrix} D_{11} & 0 & D_{13} & 0 \\ 0 & D_{22} & 0 & D_{24} \end{bmatrix} \tag{3.18}$$

Using this structure, errors in the state of one joint have no direct influence on the control signal in another joint (i.e. there are no cross terms). The Δu_P term can be seen as the proportional term of the RC, since it leads to a change in u proportional to the errors in the previous iteration. The Δu_D term can be seen as the damping term of the RC between two iterations, since it leads to a change in u proportional to the change in errors between two iterations.

3.3 | Experimental setup

We tested our approach on a two DOF SCARA type arm. This type of arm can perform industrially relevant tasks with a simple mechanical design. In the experiment, the arm has to perform a rest to rest motion, typical for a pick and place task. In this section the hardware setup and task description will be addressed, along with parameters in our approach that were set specifically for the experiments. Note however that the general approach does not depend on the exact values of the parameters, which were manually tuned.

3.3.1 Hardware setup

Fig. 3.1 shows a picture of the experimental two DOF robotic arm [131]. The arm consists of two 18x1.5mm stainless steel tubes, connected with two revolute joints, with a spring on the first joint. A gripper is connected to the end of the second tube. The motors are placed on a housing and AT3-gen III 16mm timing belts are used to transfer torques within the housing. The joints are actuated by Maxon 60W RE30 motors with gearbox ratios of respectively 66:1 and 18:1. The timing belts provide additional transfer ratios of 5:4 on both joints. The parameters of this robotic arm are listed in Table 3.1.

The large damping terms are caused by the back-emf of the motors (since we use voltage control and not current control). The viscous and Coulomb friction are neglected in this chapter since they are small compared to the back-emf induced damping. The equations of motion and the transformation to momenta can be derived using standard methods [96, Chap. 3], and are omitted from this chapter because they are too long. Because the second joint is connected to its motor via

Table 3.1: The model parameters of the two DOF arm.

Parameter	Variable	Value	Unit
Damping	μ_{v1}, μ_{v2}	7.48, 0.56	Nms/rad
Inertia	J_1, J_2	0.0233, 0.0871	kgm^2
Mass	m_1, m_2	0.809, 1.599	kg
Length	l_1, l_2	0.410, 0.450	m
Position of COM	l_{g1}, l_{g2}	0.070, 0.325	m
Motor constant	k_{t1}, k_{t2}	25.9, 25.9	mNm/A
Gearbox ratio	g_1, g_2	82.5:1, 22.5:1	rad/rad
Spring stiffness	k_1	1.6	Nm/rad

a parallelogram mechanism (see [131]), the angle of the second arm is taken as the absolute angle, i.e., relative to the world frame.

The voltage control in the brushed motors is generated as a purely feedforward signal. For brushless motors, feedback on the angles would be required to commutate the current signal. Similarly, the current signal in the previous chapter uses feedback of measured current to set the required voltage. In both these cases, the feedback loop is much faster than the dynamics of the robot arm as a whole. So it is sensible to consider these robot arms to be controlled feedforwardly.

3.3.2 Task description

We let the manipulator perform a cyclic pick and place motion, with pick and place positions at [0.4, 0.1] rad and [-0.2, -1.2] rad respectively. At the pick and place position, the arm has to stand still for 0.2s, which would be required to pick or place an object. The time to move between the pick and the place position is 1.4s. Hence, the total time of one cycle is 3.2s.

3.3.3 Optimization

To find a robustly stable trajectory an optimization is used. First, the trajectory is optimized for fast convergence, by minimizing the maximal eigenvalue modulus of the linearized Poincaré map, see the stability measure in [132]. The input was taken as a piecewise linear input, consisting of 20 pieces, with an absolute maximum value of 5V. Then, the robustness of the resulting trajectory is verified using the approach outlined in Section 3.2. The uncertain dynamics $\Delta(t)$ consist of two uncertainties: uncertainty on the linearized stiffness on the first and second joint. The bounds on these uncertainties are constant and taken as ± 0.3 and ± 1.0 Nm/rad respectively.

Table 3.2: The parameters used in optimization and RC

Symbol	Value	Symbol	Value
$t_{RCfinal}$	300 s		
P_{11}	1.5 V/rad	D_{11}	0 V/rad
P_{22}	0.6 V/rad	D_{22}	0.4 V/rad
P_{13}	0.3 Vs/rad	D_{13}	0 Vs/rad
P_{24}	0.4 Vs/rad	D_{24}	0 Vs/rad

3.3.4 Repetitive control parameters

We let the robotic arm learn the cycle from simulation while $t < t_{RCfinal}$. During this learning period, the learning gain decreased linearly from $\alpha = 1$ at $t = 0$ s to $\alpha = 0$ at $t = t_{RCfinal}$ s.

The filter gains we used are equal to

$$\phi(i) = \sigma(i) = \frac{-(i-1)(i-50)}{|-(i-1)(i-50)|} \qquad \forall i \leq 50 \tag{3.19}$$

$$\phi(i) = \sigma(i) = 0 \qquad \forall i > 50 \tag{3.20}$$

The result of using these filter gains is that the change in input signal depends on errors in steps $k-p$ to $k-p+50$, see Eq. 3.13. Therefore the RC scheme is non-causal. This filter has two purposes. First, taking a weighed average over multiple measurements reduces the influence of sensor noise. Second, the non-causality takes into account that the input relates to the acceleration, which means it takes some time to have a measurable effect on the state, which consists of the positions and the velocities.

The remaining parameters are shown in Table 3.2. All PD and filter gains we used in the repetitive control algorithm are based on experience rather than on extensive calculations. Also, note that in this chapter we do not not prove that the repetitive controller is stable. Experience on the robot shows that the repetitive controller is stable as long as the gains P_{ij} and D_{ij} are not too high.

3.4 | Results

This section presents the results from both the optimization in simulation and hardware experiments. First, using the LMI based robustness analysis, we optimize the trajectory for both stability and robustness. Second, we learn the open-loop controller that tracks this trajectory. When the controller is learned,

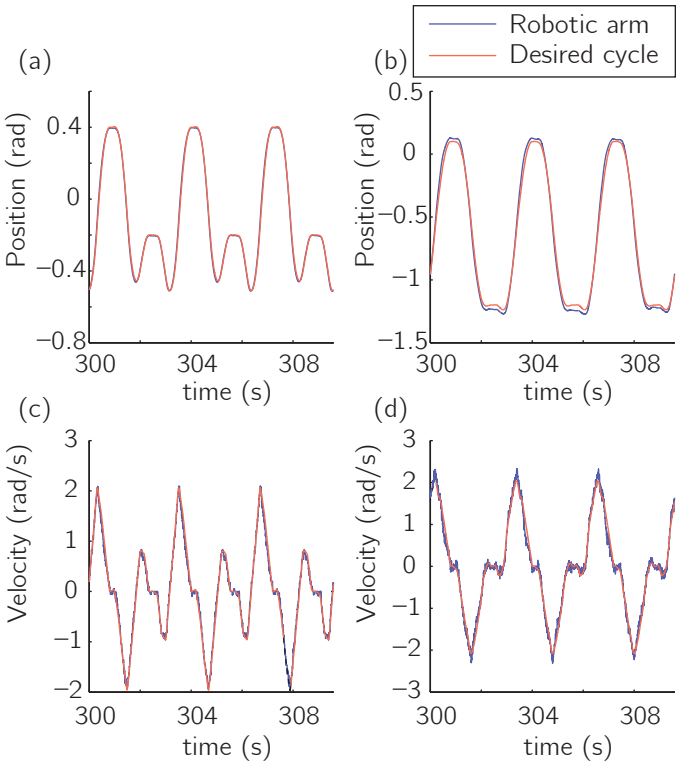


Figure 3.3: The cycle of the robotic arm after learning. The red line shows the desired cycle that was obtained from simulation. The blue solid line shows the cycle of the robotic arm after 300s of learning. (a) The position of the first joint, (b) the position of the second joint, (c) the velocity of the first joint and (d) the velocity of the second joint.

the feedback is disconnected and the robotic arms can perform its task with solely open-loop control. The video accompanying this chapter shows the hardware experiments.

3.4.1 Simulation results

The red lines in Figs. 3.3a-d show the cycle obtained from optimization. The cycle has a maximal eigenvalue modulus of 0.70 and a robustness value of $\epsilon_x = 0.038$. These results are interpreted in Section 3.5.2.

3.4.2 Hardware results

Fig. 3.5 shows the learning curve of the repetitive control. It shows two curves that correspond to the absolute position error of the end effector at the pick positions (blue line) and the place position (red line). These errors were calculated by taking

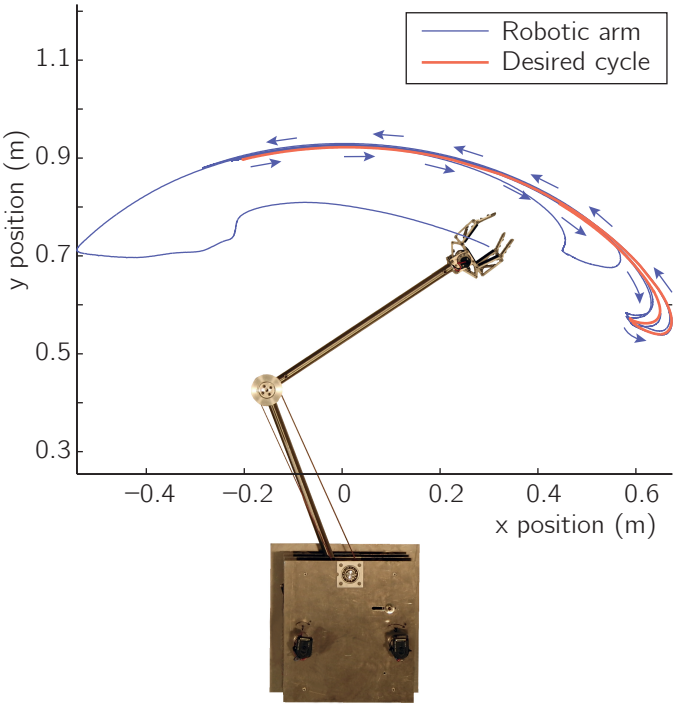


Figure 3.4: This figure shows the motion of the arm in workspace. The red line shows the desired cycle that was obtained from optimization. The blue solid line shows the motion of the robotic arm that does not start on the cycle but converges to the cycle. After approximately two cycles, the arm returns to tracking the desired cycle with its final accuracy of 2.5 cm.

the errors at the time in the cycle when picking and placing are to take place. The learning phase takes about 40 cycles, which corresponds to 128s. After the cycle is learned, the error at the pick position is approximately 1cm and the error at the place position is approximately 2.5cm. These errors are small enough to perform many basic pick and place tasks, as found for instance in the food and packaging industry. Furthermore, these errors are also within the grasping-ranges of modern robotic grippers [21][89].

The blue solid lines in Figs. 3.3a-d show the cycle after it was learned on the robotic arm. The graphs show that the cycle on the robotic arm is the same as the one obtained from simulation.

Fig. 3.4 shows the motion of the gripper in workspace. The red line shows the cycle that was obtained from simulation and the blue solid line shows the robotic arm that starts at a perturbed state and converges to the desired cycle. Within

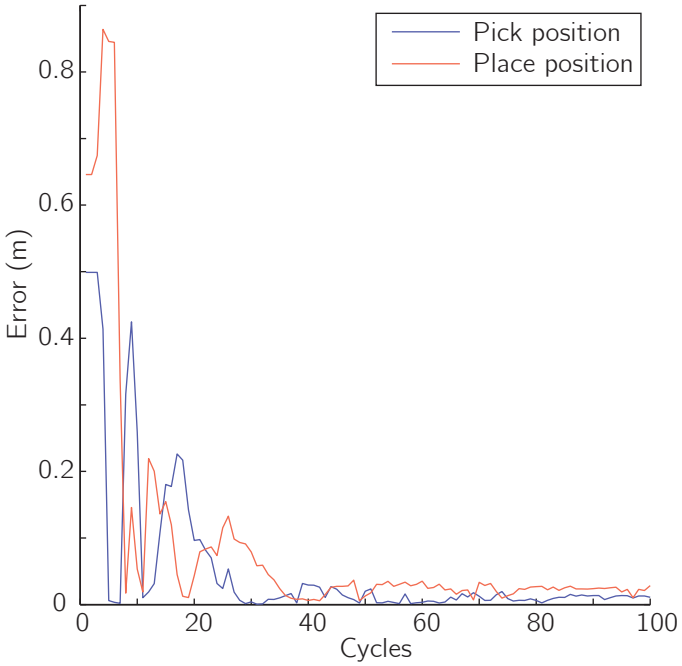


Figure 3.5: The absolute error in the pick and place positions while learning. This graph shows that the absolute error decreases to approximately 1 cm for the pick position and 2.5 cm for the place position.

two cycles, the arm follows the desired cycle.

3.5 | Discussion

3.5.1 Applicability

The techniques presented in this chapter make open-loop stable manipulation applicable. Previous work showed that model uncertainties lead to a cycle that differs too much from the intended cycle to be applicable in e.g. pick and place tasks [132]. With the learning of the open-loop controller, the errors in the position of the gripper decreased to 1-2.5cm. Furthermore, when the arm is perturbed, it converges back to the desired cycle within approximately two cycles.

Some applications, however, might require a more accurate controller (i.e. with errors smaller than 2.5 cm). There are no fundamental problems that limit the accuracy of our approach. There are however, two problems specific to our robotic arm and implementation that do limit the accuracy. Firstly, velocity estimation is

not accurate due to quantization noise due to a coarse encoder. This makes it hard to learn the desired cycle. Secondly, the state is filtered in our implementation of repetitive control, leading to a smooth control signal. However, a non-smooth control signal might be required to track the desired cycle more accurately. In conclusion, depending on the application, the setup and implementation can be changed to increase the accuracy to the desired level.

The learned feedforward approach is applicable to many tasks. A new cyclic task can be handled by the same procedure of learning a feedforward trajectory, testing the robustness with Algorithm 1, and learning the input signal with repetitive control. At the moment the learning takes the most time, 300s. The optimization (1s) and robustness test (5s) take much less time. Reducing the time needed to learn a new task will further increase the applicability and is therefore an important part of our future research.

The main drawback of the current approach is that some feedback is required during the learning phase. However, this feedback can be delayed (almost a whole cycle) and can be removed after the learning has finished. The (delayed) feedback could therefore be provided by e.g. cameras. One interesting direction for future work would be to research repetitive control schemes without full state feedback. This could mean omitting certain states or reducing the frequency of the feedback.

This feedback requirement also comes up in situations where large disturbances are expected to occur after learning. In such situations the robot should know whether it has converged back to the desired cycle. This requirement can be satisfied with cameras, or even more basically, by using a switch at a certain position that checks the timing of passing that position.

3.5.2 Interpreting robustness measure

The robustness test depends on the choice of Δ, which is to some degree arbitrary. In previous experiments on this robotic arm, bending of the second joint was hypothesized to be one of the main causes of model-reality mismatch [132]. Therefore Δ was chosen to emphasize this error, which depends only on the position of the second joint.

For the chosen Δ, the resulting robustness measure ϵ_x of 0.038 indicates that the spring stiffness of ± 0.038 Nm/rad could be added to the second joint. This seems insignificant to the effective spring stiffness of around 1.6 Nm/rad that is in place on the first joint. However, note that adding any negative spring stiffness to the second joint would make that joint unstable by itself. Therefore low values of ϵ_x should be expected.

3.5.3 Hybrid systems and Coulomb Friction

Despite the fact that robustness against modeling errors is checked, there are still two modeling-related issues that are not accounted for in the current approach. Firstly, when performing an actual pick and place motion, the model will consist of two phases with different end-point masses with an impact phase in between to model the grasping. Secondly, a more accurate friction model would include Coulomb friction. Both these effects can be added by using a hybrid system model. Such hybrid models are quite generally used in analysis of systems with limit cycles, mostly to cope with impact in walking, e.g. [180]. Under some basic transversality conditions, as discussed in [24], the current approach can be extended to such hybrid models. We see that inclusion as the logical next step in the research in open-loop control of robotic arms.

3.5.4 Scalability

The experiments were done on a two DOF robotic arm. Such an arm has similar dynamics to SCARA type arms, which are often used in industry. The current approach therefore already can be used for realistic industrial situations. However, there are also many robotic arms which have a larger number of joints and rotate in 3D. For such a setup the approach remains untested.

There are three possible concerns for this approach for higher DOF robots. Firstly, there is the possibility that no stable trajectories exist. This is unlikely, and it can be ensured not to happen if springs are used on all joints to stabilize the system. Secondly, the required computation will become more extensive. However, because only a feedforward signal is required, no full state exploration is necessary. Furthermore, the number of free parameters in the robustness approach is independent of the number of DOFs on the robot. Thirdly, although earlier results on a two DOF arm suggest that the basin of attraction of our approach is quite large [132], for multiple degrees of freedom it becomes more likely that the robotic arm will converge to a different cycle after a large perturbation. Again, this could be prevented by using springs on all joints to stabilize the system. Combining all of the above, we expect that our approach scales well to higher DOF robotic arms.

3.6 | Conclusion

In this chapter we showed that repetitive tasks can be performed stably by robotic arms with an open-loop controller, even when an accurate model is not available. We used an LMI-based robustness analysis to check the robustness to model inaccuracies. We then used a repetitive control scheme to track this trajectory with the robotic arm. Hardware experiments show that using this approach, position

errors can be reduced to 2.5 cm, making open-loop control applicable in tasks such as picking and placing objects.

Acknowledgement

This work is part of the research programme STW, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO).

4

Sensor-free manipulation: Extending robustly stable feedforward control to hybrid systems

Wouter Wolfslag, Michiel Plooij, Robert Babuška and Martijn Wisse,
This chapter was not published before.

Abstract

Robotic arms perform their tasks by relying heavily on sensor feedback, which can be inaccurate or even unavailable. Furthermore, acting upon feedback is always a delayed and reactive behavior. As such, this reliance on feedback is not always the desired control strategy. When it is not, a more human like, feedforward based, control strategy should be used. In order to study these feedforward strategies, the goal of this chapter is to make a robotic arm perform a basic pick-and-place task using feedforward control only. Despite their simple descriptions, pick-and-place tasks should be modeled as hybrid systems, as the dynamics of the robot arm change when grasping or releasing its payload. Furthermore, when performing the task without sensors, two challenges need to be overcome: stability and robustness against modeling errors. A feedforward control scheme that handles both these challenges on a hybrid system does not exist yet. In this chapter we extend a robustly stable feedforward control scheme to hybrid systems. We used this scheme to compute robustly open-loop stable cycles, after which a three-degree-of-freedom robotic arm learned to follow these cycles using Repetitive Control. The results show that the cycle was tracked with maximum position errors of 2.5 cm at the pick and place positions. For as far as we know, this is the first time that a robotic arm robustly performs a pick and place task by solely using the most low-level open loop controller possible: a voltage on the motors that is only a function of time.

4.1 | Introduction

One of the most basic tasks that both humans and robots perform is a pick-and-place task: in an uncluttered environment an object is to be moved between a relatively small number of positions. This task is fundamental because of its simple description, but also because of its potential for automation in industry. Some of this potential has already been realized, but there is still a large gap in quality of motion between robots and humans. The difference in motion quality largely comes about due to the clever motion strategies humans use.

One human strategy is the extensive use of feedforward control [42], which greatly improves the accuracy and speed of human movements. Incorporating a feedforward term in robot control is known to reduce sensitivity to sensor noise [91] and to increase bandwidth. We argue that more human-like feedforward strategies should be applied (along with feedback) in robotics, to obtain better performance. To

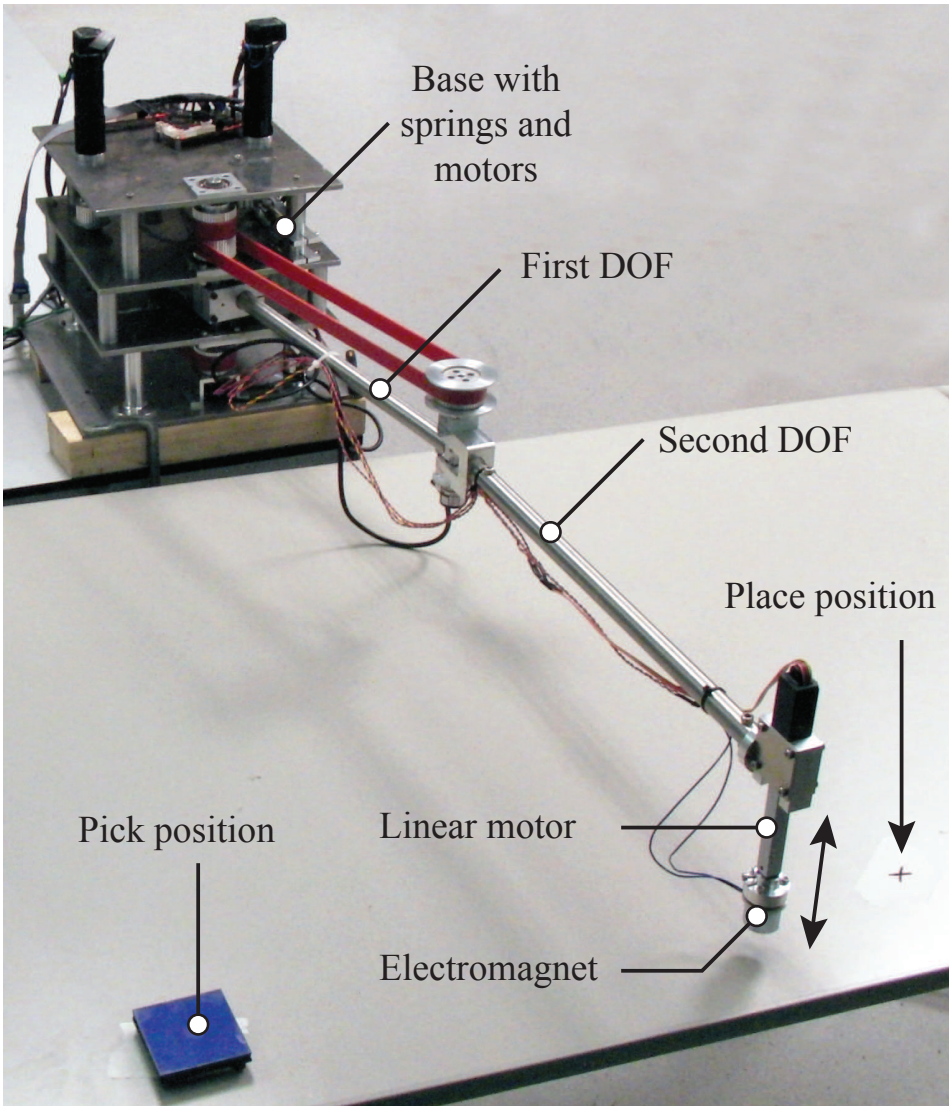


Figure 4.1: A photograph of the experimental setup. The task is to move the object from its current position, the pick position, to the indicated place position

focus on understanding the capabilities of feedforward controllers from a robotics perspective, we take an extreme approach in this chapter: we use no feedback at all.

Such a study is not only interesting for obtaining insights in feedforward control, it

also has the potential to directly increase the effectiveness of robots. Specifically, if a robotic manipulator performs simple tasks in a relatively structured environment with known pick and place locations, a design without feedback can be beneficial. No feedback means no need for sensors or high-rate sensing electronics, allowing the robotic arm to be simpler and cheaper. Furthermore, in situations where feedback is not reliable or when it is even impossible, for instance due to small size, vacuum, or radiation, feedforward control is the only solution.

The theoretical interest along with the direct benefits have made purely feedforward controlled systems a well appreciated research topic. This research has focused on two main problems: stability and modeling errors. The stability of open loop systems has mostly been studied using limit cycles. Within robotics, examples are the work on passive walking robots by McGeer [109], subsequent feedforward controlled walkers by Mombaur et al. [113, 114] and a juggling robotic arm by Schaal and Atkeson [146]. Plooij et al. [132] showed that pick-and-place motions can also be performed as a feedforward-controlled stable limit cycle. The capability of a purely feedforward-controlled robot to handle modeling errors has also been studied. Firstly, Singhose et al. [155, 156] researched vibration reducing open-loop controllers while taking into account the robustness to uncertainty in the natural frequency and damping of the system. Secondly, Becker and Bretl showed that it is possible to steer a differential drive robot to a desired position, even when the wheel diameter is uncertain [11]. And finally, Plooij et al. [128] showed that reaching movements can be performed with feedforward control despite model inaccuracies.

A purely feedforward controlled robot needs to handle both stability and modeling errors at the same time, that is, it needs to be robustly stable. One way to achieve this robust stability is to obtain mechanical feedback, by using very stiff design elements, such as cam-systems, end-stops or stepper motors. Such stiff design elements must resist large forces, and therefore come with a downside: they must be large and heavy.

Such heavy elements can be avoided, if we find a feedforward controller whose robustness relies on choosing suitable trajectories. A first step in this direction was made by Wolfslag et al. [185] who proposed a learning approach. With this approach, the inputs to movements that are stable even with model uncertainties, were learned online. Very slow feedback was used during learning and it was removed afterwards. With this approach, it can be ensured that a trajectory that performs well in simulation is executed stably and accurately on the hardware.

The problem with this approach is that it does not allow for hybrid systems, whereas for many real world tasks the robot has to be modeled by a hybrid system. For instance, in a pick-and-place task, the inertia of the robotic arm changes due

to picking up an object. Furthermore, the picking and placing actions require a different model, possibly including impact. As a result, open-loop pick-and-place performance on the robot hardware could not be ensured from planning in simulation.

Therefore, the main contributions of this chapter are the extension of the approach from [185] to allow for hybrid systems and to verify that extension by performing a pick-and-place task on a purely open loop controlled robot.

The rest of this chapter is structured as follows. First, we explain our methods in Section 4.2. This includes the experimental setup, the robustness analysis, our new extension of this analysis to hybrid systems and the Repetitive Control scheme we used. Next, in Section 4.3, we present the results we obtained. We then discuss these results in Section 4.4 and we conclude this chapter in Section 4.5.

4.2 | Methods

In this section we explain the methods used. We start with explaining the robotic arm and the task it has to perform. Next, we explain the stability and robustness analysis, which is the same as in Chapter 3. Then, we extend this analysis to hybrid models, allowing the robotic arm to actually pick and place objects. And finally, we explain the Repetitive Control scheme designed to track the desired trajectory.

4.2.1 Experimental setup

Fig. 4.1 shows a picture of the experimental three DOF robotic arm [131] used in this research to perform pick-and-place tasks with an open-loop controller. A schematic drawing of this setup is shown in Fig. 4.2. The arm consists of two 18x1.5mm stainless steel tubes, connected with two revolute joints, with a spring on the first joint. A Firgelli L16P linear motor is connected to the end of the second tube and an electro magnet is connected to the linear motor as a gripper. The motors for the first two DOFs are placed on a housing and AT3-gen III 16mm timing belts are used to transfer torques within the housing. The joints are actuated by Maxon 60W RE30 motors with gearbox ratio of respectively 66:1 and 18:1. The timing belts provide an additional transfer ratios of 5:4 on both joints. On both joints, a spring with a stiffness of 1.6 Nm/rad is connected in parallel with the motor. These springs help to stabilize the robot. In particular, the spring on the first joint prevents errors on that joint from accumulating, and is necessary to be able to obtain stable feedforward controllers[132]. The parameters of this robotic arm are listed in Table 4.1.

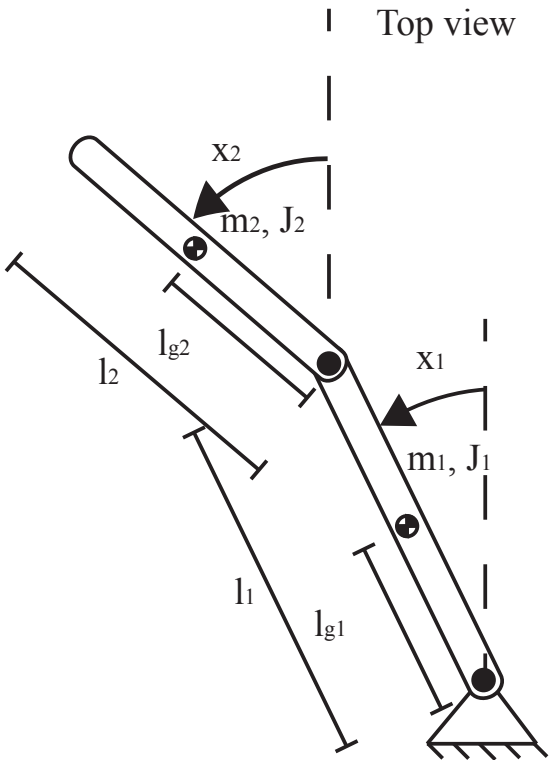


Figure 4.2: A schematic drawing of the experimental setup.

Table 4.1: The model parameters of the two DOF arm.

Parameter	Symbol	Value	Unit
Damping	μ_{v1}, μ_{v2}	7.48, 0.56	Nms/rad
Inertia	J_1, J_2	0.0233, 0.0312	kgm^2
Mass	m_1, m_2	0.809, 0.784	kg
Length	l_1, l_2	0.410, 0.450	m
Position of COM	l_{g1}, l_{g2}	0.070, 0.194	m
Motor constant	k_{t1}, k_{t2}	25.9, 25.9	mNm/A
Gearbox ratio	g_1, g_2	82.5:1, 22.5:1	rad/rad
Spring stiffness	k_1, k_2	1.6, 1.6	Nm/rad

The large damping terms are caused by the back-EMF of the motors (since we use voltage control and not current control). Friction is neglected in this chapter since it is small ($< 10\%$) compared to the back-EMF induced damping. The equations of motion and the transformation to momenta can be derived using standard methods [96], and are omitted from this chapter because they are too long. Because the

second joint is connected to its motor via a parallelogram mechanism (see [131]), the angle of the second arm is taken as the absolute angle (i.e. relative to the world frame).

The task the arm has to perform is a standard pick-and-place task in which it has to transfer packages with a weight of 168 g from the pick to the place position. The pick and the place positions are (in terms of positions of the two rotational joints) [0.4, 0.2] rad and [-0.4, -0.3] rad respectively. At those positions, the arm stands still for 0.6 s to pick or place a package.

4.2.2 Robustly stable feedforward control

To perform a pick-and-place task using only feedforward control, the motion needs to be stable and accurate. To achieve these requirements, an approach described in [185] is used. This approach is briefly explained in Sections 4.2.2 to 4.2.3, and summarized in Algorithm 2. In Section 4.2.4, we then show how to extend that approach to incorporate hybrid systems, allowing us to study a pick and place task.

A mechanical system using generalized positions and momenta as state space variables can be described by the differential equation $\dot{x} = f(x) + Bu$. It can be linearized along a trajectory $x^*(t)$ caused by input $u^*(t)$:

$$\frac{d\bar{x}}{dt} = A^*(x^*(t))\bar{x} = A^*(t)\bar{x} \tag{4.1}$$

with

$$A^*(t) = \left.\frac{\partial f}{\partial x}\right|_{x^*(t)} \tag{4.2}$$

where $\bar{x}(t) = x(t) - x(t)^*$ is the state error. For the ease of notation, the time dependency of variables is occasionally dropped if it is unambiguous to do so. For example $A^*(t)$ will be written as A^*.

Note that the linearized stability depends on $A^*(t)$ over time, which is independent of the input $u^*(t)$. This independence will prove useful and it is the reason for choosing momenta instead of velocities in the state. When there are model uncertainties, the above equations become invalid for two reasons. First $A^*(x^*(t))$ is no longer accurately known when in state $x^*(t)$. Secondly, when using a fixed open-loop controller on an uncertain system, the trajectory is not fully predictable, so in general $x(t) \neq x^*(t)$, when using the input $u^*(t)$. To solve these two issues, a two-step approach is used:

1. Analyze the robust stability of the system defined by $A^*(t)$, i.e. add an uncertain term to $A^*(t)$ and check whether the system is still stable.

2. Learn the input that makes sure the system follows the trajectory $x^*(t)$. Recall that the input does not influence the stability of the linearized system. Therefore, the stability of the motion, as determined in the first step, is still valid even for the learned input.

4.2.3 Robust stability analysis

In this section, the robustness test from [185] is briefly explained. The test aims at finding a Lyapunov function that works for a system with added uncertainty. To find the Lyapunov function, a transformation into a time invariant system is used. As the resulting numerical scheme can be sensitive to numerical noise in highly stable system modes, the system is reduced by assuming the error in those modes is always 0.

The first step in analyzing the robust stability is to add uncertainty to the system. We model the uncertainty of the system through an integer number of uncertain parameters δ_j, that enter the linearization affinely, using known state-dependent and possibly time-dependent matrices $\Delta_j(t)$:

$$A^*(t) = \frac{\partial f}{\partial x}(t) + \sum_j \Delta_j(t)\delta_j \tag{4.3}$$

where we take $A(t) = \frac{\partial f}{\partial x}(t)$ as the certain part of the dynamics, and $\Delta(t) = \sum_j \Delta_j(t)\delta_j$ as the uncertain part. As a result, we have a time varying uncertain system $\dot{\bar{x}} = (A(t) + \Delta(t))\bar{x}$.

Now we aim to find a quadratic Lyapunov function, $J = \bar{x}^T M(t)\bar{x}$, with positive definite $M(t)$ for this system. For cyclic systems, stability is verified if an $M(t)$ can be found that satisfies the following constraints:

$$MA + \dot{M} + A^T M + \epsilon_x \Delta^T M + \epsilon_x M\Delta \prec 0 \qquad \text{C5}$$

$$M_f - M_0 \succ 0 \qquad \text{C6}$$

where $\prec$ and $\succ$ are used to indicate negative/positive definiteness respectively, $\dot{M}$ is the time derivative of M, the subscripts 0 and f denote initial/final time and $\epsilon_x > 0$ is used to scale the uncertainty. The largest ϵ_x for which these constraints can be made to hold is a robustness measure, as it indicates how large the uncertainty can be while the quadratic Lyapunov function still guarantees stability.

Testing a specific $M(t)$ for feasibility with respect to the constraints directly is not computationally tractable. To turn the constraints into a computationally tractable test, two steps are required. The first step is derived from the theory of Linear Matrix Inequalities [150]: we know that if each δ_j is constrained to some interval $\delta_j \in [\underline{\delta}_j, \bar{\delta}_j]$, then constraint C5 holds for all $\Delta(t)$, if it holds for the

$\Delta(t)$ created by the vertices of the hypercube of the allowed δ_j. The $\Delta(t)$ values connected to these vertices will be called $\Delta_0(t)$, and they form a finite set. The choice for $\Delta_0(t)$ depends on the expected model inaccuracies. The second step is a discretization over time. Tobenkin et al. [171] studied the correctness of such a discretization for a similar stability test. They showed that for a given infeasible Lyapunov function, there exists a sampling time that is small enough, such that its sampled analogue is found infeasible as well. As such, if the sampling time is small enough, a Lyapunov function that is tested to be feasible indeed gives a proof of stability. Unfortunately, it is unknown how to find a sampling time that is guaranteed to be sufficiently small. We use a step size of 0.001 s in our discretization, which we expect to be sufficiently small.

Now that we have a method to verify whether a specific $M(t)$ satisfies the constraints, what remains is a way to find $M(t)$. In order to compute a (suboptimal) $M(t)$, we use a method that is based on the fact that all time varying systems with periodic coefficients can be transformed into a time invariant system [53]. The transformation to the time invariant system is based on the state transition matrix Φ of the nominal system description $\bar{x}(t) = \Phi(t)\bar{x}(0)$, which can be found by integrating the following variational equation:

$$\dot{\Phi}(t) = A^*(t)\Phi(t), \quad \Phi(0) = I \tag{4.4}$$

Now if we define $L(t) = \Phi(t)\Phi(t_f)^{-\frac{t}{t_f}}$ and the transformation $\bar{x} = L(t)y$, we get for the state transition matrix of y:

$$\dot{y} = \frac{1}{t_f}\ln(\Phi(t_f))y \tag{4.5}$$

which is a time invariant, but possibly complex-valued system [53]. For this system we can then find a Lyapunov function $y^T M_y y$ by solving for the positive definite matrix M_y using standard LMI techniques. For the numerical implementation of these techniques we use YALMIP [100] which uses SeDuMi [163]. We optimize M_y such that the Lyapunov function $y^T M_y y$ has the smallest time derivative, i.e. maximize ϵ_y in:

$$\frac{1}{t_f}\ln(\Phi(t_f))^T M_y + M_y\frac{1}{t_f}\ln(\Phi(t_f)) + \epsilon_y I \prec 0 \tag{4.6}$$

The resulting M_y is then transformed back to the $\bar{x}$ coordinates:

$$M(t) = L(t)^{-T} M_y L(t)^{-1} \tag{4.7}$$

Since this construction leads to a periodic $M(t)$, constraint C6 holds. To verify the robustness of the system, the maximal value of ϵ_x for which C5 holds is found, using the discretization over time and the hypercube of allowed uncertainties Δ_0.

In many cases one or more of the singular values of the state transition matrix $\Phi(t_f)$ are nearly zero [144]. This means that some errors are reduced to nearly zero after one cycle. Unfortunately, the computation of the robustness measure is numerically sensitive to these near singularities, even though the corresponding errors are normally insensitive to relatively small parameter changes. Therefore, we choose to filter (project) out the directions in which these nearly singular errors occur, reducing the dimensionality of the system. An approach in which the near singular directions are regularized, for example by modifying ridge regression, could also be possible. The projection approach was chosen here, as the effect on the system under study is more easily interpreted.

The filtering process consists of two transformations, which together result in a lower-dimensional system on which the robustness test can be performed. The first transformation is based on the singular value decomposition of the final stability matrix: $\Phi(t_f) = U\Sigma V^T$. We set a cut-off ratio for the singular values, meaning we only look into the directions with singular value higher than that cut-off ratio. This ratio is set to 10^{-4} in our experiment. We define U_χ as the columns corresponding to the singular values higher than the cut off ratio. U_χ forms a basis for the space of errors with associated singular values higher than the cut-off ratio.

To track only the not nearly singular errors, we consider the system in a coordinate frame as defined by the columns of U. The error in that coordinate frame will be called $\bar{U}$ and can be found using the transformation: $\bar{U} = U^{-1}\Phi(t)U_\chi$. Note that this transformation does not reduce the dimension of the system description. To do so, we want to move the coordinate system along with $\bar{U}$, such that the only errors taken into account are the errors in the non-singular directions, while at the same time the coordinate system is periodic in time. To create such a moving coordinate system, we use a coordinate system that is spanned by the following matrix:

$$R = \text{r.c.e.f.}(\bar{U}) \tag{4.8}$$

with r.c.e.f. denoting the reduced column echelon form operator. This operator maps a matrix to its reduced column echelon form, that is, it takes the transpose of the, more familiar, row echelon form of the transpose of a matrix. The coordinate system found this way defines a lower-dimensional description of the error (which we call Z) in the non singular directions:

$$Z = R^\dagger \bar{U} \tag{4.9}$$

where the † symbol is used to indicate a pseudo inverse.

This Z-description of the error can be used in the robustness scheme. This requires the transformation of the system matrices A and Δ_0 matrices to the Z-space.

Doing so defines the time derivative of Z as $\dot{Z} = (A_Z + \Delta_Z)Z$ with

$$A_Z = \dot{R}^\dagger R + R^\dagger U^{-1} A U R \tag{4.10}$$

$$\Delta_Z = R^\dagger U^{-1} \Delta_0 U R \tag{4.11}$$

Note that the dimension reduction outlined above introduces non-conservativeness when transforming the error caused by Δ_0. When the system dynamics change from the nominal value, the actual error is no longer in the space spanned by the non-singular solutions of the nominal system. The actual error is therefore not in the lower-dimensional space described by Z. The pseudo inverse in the definition of Z projects this error back, setting the nearly singular errors to 0 at every time step. The combined effect of all these errors in the singular directions is not per definition negligible, meaning the robustness measure is an estimate, not a guarantee. To partially address this issue, we take the worst case effect of Δ_0 in all directions, and add that to all the non-singular directions:

$$\tilde{\Delta}_Z = \Delta_Z + ||R^\dagger U^{-1}||_\infty \cdot ||\Delta_0 U R||_\infty \cdot I \tag{4.12}$$

We use this model error in the final form of the robustness constraint:

$$\dot{M} + A_Z^T M + M A_Z + \epsilon_x \tilde{\Delta}_Z^T M + \epsilon_x M \tilde{\Delta}_Z \prec 0 \tag{4.13}$$

The above equation requires the computation of the time derivative of the column echelon form, which we do with a finite difference scheme. The complete computation is summarized in Algorithm 2.

Algorithm 2 Robustness test

procedure ROBUSTNESSTEST($x^*(t)$,Δ_0)
- Linearize system around trajectory ▷ Eq.(4.2)
- Perform dimension reduction ▷ Eqs.(4.9)-(4.12)
- Transform system to time invariant system ▷ Eq. (4.5)
- Find Lyap. func. for time invariant system ▷ Eq. (4.6)
- Initialize robustness measure, $\epsilon_x \leftarrow 1000$
- **for all** t in time discretization **do**
  - Transform Lyap. func. to reduced system ▷ Eq. (4.7)
  - $\bar{\epsilon} \leftarrow$ maximum ϵ_x to satisfy Eq. (4.13) at time t
  - $\epsilon_x \leftarrow \min(\bar{\epsilon}, \epsilon_x)$

return ϵ_x

4.2.4 Extension to hybrid systems

To perform a real world task, interaction with the environment is required. Such an interactive task can often be described as a system with multiple phases each

with different dynamics. For instance, in the case of a pick-and-place task, a motion consists of four phases: moving without object, grasping, moving with object, releasing; each with different dynamics. A simple model makes grasping and releasing instantaneous, meaning the dynamics of the robotic arm switch from no load to carrying the object instantly. Such a switch in general also influences the state, which means it is modeled as a discrete event, such that the state directly after the switch is a function of the state directly before the switch.

A system that has multiple phases, switching between them at set points in time or space, is called a hybrid system. Such systems can be described in many ways. The description below gives a class of hybrid systems for which the robustness test can be used. First, the state space, including time, is divided into regions $\mathcal{D}_i$. These regions have boundaries $\mathcal{S}_i$. Then the following equations of motion describe the system.

$$\dot{x} = f_i(x) + B_i u, \qquad x \in \mathcal{D}_i \wedge \notin \mathcal{S}_i \tag{4.14}$$

$$x^+ = s_i(x^-), \qquad x^- \in \mathcal{S}_i \tag{4.15}$$

Where x^- and x^+ are the states at the beginning and end of the instantaneous switching step, and $f_i(x)$ and $s_i(x^-)$ are the continuous and switching dynamics of the system at hand.

In order to test for robustness, the system must be linearized around a trajectory. The variational equation (4.4) can still be used to find the state transition matrix, by using a discrete step at the switches. For these switches, care must be taken to take into account that the switches of the perturbed trajectory will not occur at the same time and state for different values of the error, see [44] for the correct equations. This linearization is only correct if the order of the phases is preserved when the trajectory is perturbed. For our description, this occurs when two criteria are met. First, the functions $s_i(x)$ should be differentiable at the switching points. This prevents the system from switching to altogether different phases. Secondly, the trajectory should be transverse to the switching boundary at the instant of switching [24]. This prevents a switch from being missed.

The robustness test of the continuous phases of the hybrid system can now be performed just as before. Note that when analyzing the continuous phase before the switch, the error at the switching time should be taken as the error the instant before switching. When analyzing the continuous phase after the switch, we should take the error the instant after switching. That way, the discrete state transition at the switch does not influence the analysis of the continuous phases.

The switching function $s_i(x)$ itself can be uncertain too. In that case, the worst case switch should be computed, given $\Phi(t_{switch})$, and those worst cases should be used in determining the state transition matrix after the switch. However, in

many applications, including ours, the switching transitions are inelastic collisions, which do not contain uncertainty.

To apply the robustness test to the pick-and-place robot, the robot must be modeled according to the hybrid system description above. The model consists of two regions, separated by the boundary $t = t_{switch}$. As the system is cyclic, there is another boundary at $t = t_f$. The continuous dynamics in the phases only differ in the end-effector mass, which is includes the object mass when $t_f > t > t_{switch}$. The first transition models grasping as an inelastic collision, setting the link velocities to zero. Therefore, the switching function is $s_i(x) = S_1 x$, with S_i is a block-diagonal matrix with blocks I and 0. The second transition (at t_f) models the releasing of the object. This release has no effect on the state of the robot, hence $s_2(x) = x$.

Combined with the extension to hybrid systems, Algorithm 2 explains a way to estimate the robustness of a given trajectory. In order to perform a pick-and-place task, we first determine a trajectory, and then estimate the robustness of that trajectory. If the robustness estimate is much larger than the expected mismatch between model and robot, we can assume the feedforward controller will lead to a stable trajectory on the robot.

4.2.5 Application in experiment

To determine the trajectory for the robot experiment, an optimization with respect to another goal than robustness is used. Specifically, the trajectory is optimized for fast convergence by minimizing the maximal absolute eigenvalue of the linearized Poincaré map, see the stability measure in [132]. Fast convergence is important for feedforward controlled robotic manipulation, as the robot will fail to perform its task while it has not converged to its desired trajectory after an error. Therefore fast convergence is more important than other performance measures such as energy efficiency or speed. The input was taken as a piecewise linear voltage, consisting of 20 pieces, with an absolute maximum value of 5V.

For our experiments, the uncertain dynamics $\Delta(t)$ consist of two uncertainties: uncertainty on the linearized stiffness of the second joint and linearized damping of the first joint. The bounds on these uncertainties are constant and taken as ± 1 Nm/rad and ± 1 Nm/rads respectively. These bounds are higher than the expected model mismatch which is in the order of 0.1 Nm/rad or 0.1Nm/rads respectively.

4.2.6 Repetitive control

The optimization and robustness verification scheme combines to be the first step in the robustly stable feedforward approach. The second step is to learn the input that tracks the trajectory found in the first step. We use Repetitive Control (RC) [98] for this learning step. RC teaches the robot to track the cyclic trajectory

accurately, despite model errors that repeat exactly every cycle. After learning, the feedback is disconnected and the task is performed stably with the learned open-loop controller. The Repetitive Control scheme is similar to that in [185]. It runs on a real time target with a sampling frequency of 250 Hz. Therefore, we will use discrete time notation with time step k.

The repetitive control scheme we used is

$$u(k) = u(k-p) + (\Delta u_P(k) + \Delta u_D(k)) \tag{4.16}$$

with

$$\Delta u_P(k) = \sum_{i=0}^{p-1} \phi(i) \cdot P \cdot \bar{x}(k-p+i) \tag{4.17}$$

$$\Delta u_D(k) = \sum_{i=0}^{p-1} \sigma(i) \cdot D \cdot (\bar{x}(k-p+i) - \bar{x}(k-2p+i)) \tag{4.18}$$

where $u(k)$ is the control input, and p is the number of time steps in one cycle. $\phi(i)$, $\sigma(i)$, P and D are explained below.

The filter gains $\phi(i)$ and $\sigma(i)$, for $i = 1, \ldots, p$, determine how much the different errors in the previous iterations contribute to the change in the input signal. The filtering accounts for measurement noise and prevents oscillations in the RC. The sum of the elements of $\phi(i)$ and the sum of the elements of $\sigma(i)$ are both equal to 1, to obtain a (weighted) moving average filter. The filter gains we used are equal to

$$\phi(i) = \sigma(i) = \frac{-(i-1) * (i-50)}{|-(i-1) * (i-50)|} \qquad \forall i \leq 50 \tag{4.19}$$

$$\phi(i) = \sigma(i) = 0 \qquad \forall i > 50 \tag{4.20}$$

The learning gains P and D are $N_u \times N_s$ matrices (with N_u the number of inputs and N_s the number of states). The P and D we used have the following structures:

$$P = \begin{bmatrix} P_{11} & 0 & P_{13} & 0 \\ 0 & P_{22} & 0 & P_{24} \end{bmatrix} \tag{4.21}$$

$$D = \begin{bmatrix} D_{11} & 0 & D_{13} & 0 \\ 0 & D_{22} & 0 & D_{24} \end{bmatrix} \tag{4.22}$$

The values for those gains are listed in Table 4.2.

Table 4.2: The parameters used in Repetitive Control

Symbol	Value	Symbol	Value
$t_{RCfinal}$	100 s		
P_{11}	1.5 V/rad	D_{11}	0 V/rad
P_{22}	0.6 V/rad	D_{22}	0.4 V/rad
P_{13}	0.3 Vs/rad	D_{13}	0 Vs/rad
P_{24}	0.4 Vs/rad	D_{24}	0 Vs/rad

4.3 | Results

In this section we show the results we obtained both from simulation and hardware experiments. The cycle that was obtained from optimization is shown in Fig. 4.3 (the red dashed line). The maximum eigenvalue of this cycle is 0.56. The robustness estimate ϵ_x is 3.17, which means that the maximum allowable model uncertainty is 3.17 times as large as the bound we set on the expected model uncertainties.

Fig. 4.3 also shows the learned cycle on the robotic arm (blue solid line). It shows that the cycle was learned reasonably well: the maximal deviation from the desired trajectory is 0.1 rad. This largest errors occurs very briefly, and is found in the second DOF. At $t = 147$ s, a manual disturbance is applied to the arm. The cycles after this disturbance show the convergence of the arm towards the learned cycle. Although the arm was only fully converged after four cycles, it never missed picking up an object in this example. Usually with a large disturbance, the arm misses two or three packages after the disturbance.

Fig. 4.4 shows the convergence of the end effector position due to the Repetitive Controller. The learning phase lasts 100 s, which corresponds to approximately 26 cycles. After this learning phase, the controller is completely open-loop. The figure shows that the error at neither the pick nor the place position increases after the learning phase has ended. Only when manual disturbances are applied at the 38th, 43rd and 50th cycle, the error increases temporarily. The disturbance at the 38th cycle corresponds to the disturbance indicated in Fig. 4.3. The maximum position error of the end-effector at both the pick and the place position after convergence is approximately 2.5 cm.

4.4 | Discussion

The extension to hybrid systems allowed the analysis to be used for a task with physical interaction with the environment. The hybrid extension can also be used to incorporate Coulomb friction, which would cause a switch in dynamics at zero velocity. Therefore, this extension is a large step in making the robust feedforward

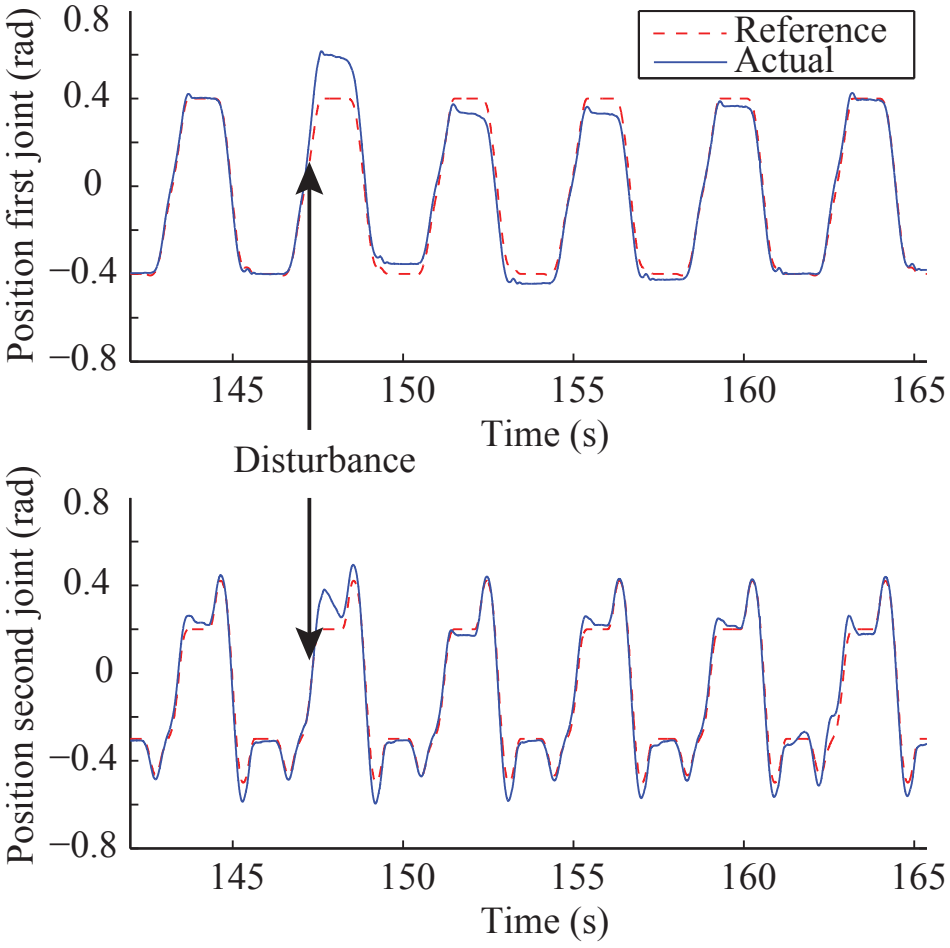


Figure 4.3: The position of the first and second DOFs as functions of time, after learning has completed. Both the desired position and the actual position are displayed. At t=147 s, a manual disturbance is applied, after which the robotic arm converges back to its cycle using only feedforward control.

control scheme introduced in [185] applicable for robots performing meaningful tasks.

After the cycle was learned on the robotic arm, the position errors at the pick and place positions were smaller than 2.5 cm. This is accurate enough as to be applicable for many practical pick-and-place tasks, for instance in the food or packaging industry. After a disturbance, the robotic arm returns to its desired path within two or three cycles, after which it can perform its task again. Applying this feedforward scheme to scenarios where an occasional delay of two or three cycles is

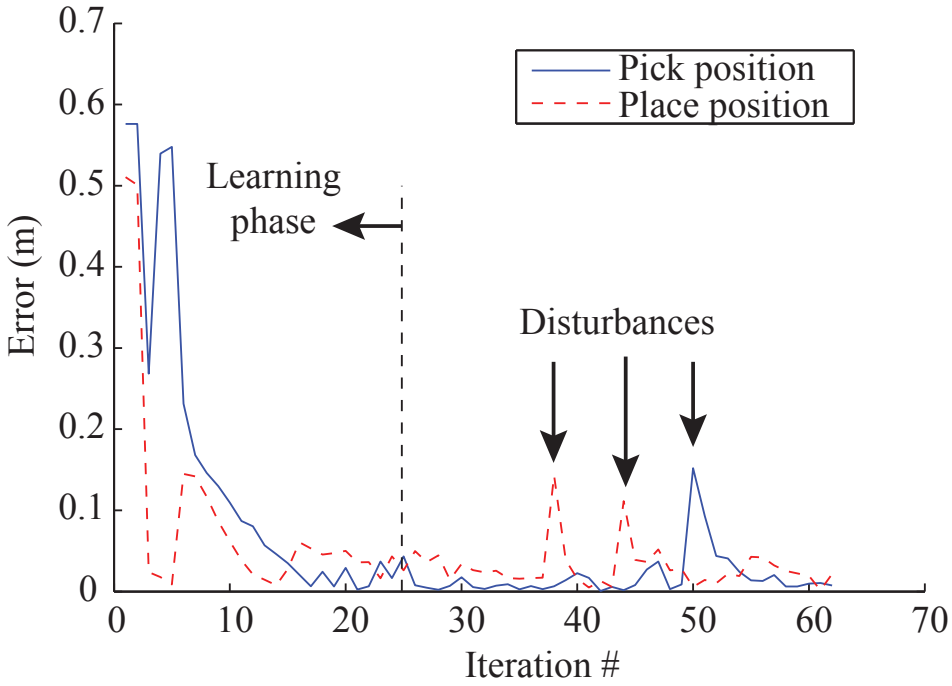


Figure 4.4: The error at the pick and the place position as function of the number of cycles that have passed. The learning phase lasts 100 s, which corresponds to approximately 26 cycles. At the 38th, 43rd and 50th cycle, manual disturbances are applied to the arm.

acceptable could reduce costs by leaving out sensors and sensor signal processing electronics.

There are two ways to make the feedforward scheme more applicable directly. The first is to decrease the time it takes to learn the feedforward signal. The current learning time is 100 s. For many applications multiple trajectories are required, which means that all those trajectories have to be learned. The goal would be to learn every cycle within a couple of iterations. The second way to improve applicability is to improve the accuracy. This could be handled by implementing some basic sensing, such as a coarse encoder. Disregarding feedback, the limitation on accuracy is caused by errors that do not repeat exactly each cycle. Such errors arise from backlash and elasticity in the mechanical design, and sensor noise during learning. These two issues are therefore important in the design of a purely feedforward controlled robotic manipulator. Finally, if the accuracy is only important at the pick-and-place positions, the RC could be tuned to emphasize these positions.

The robustness measure in this chapter has two directions of improvement. First,

the robustness measure potentially overestimates the robustness of the robot model, due to the discretization step and the dimension reduction steps. This means that there are no hard guarantees that the control scheme will work on the physical robotic arm. The estimate on the robustness gives a large enough margin to assume that the actual model uncertainties will not lead to an unstable system. A conservative robustness measure would allow this assumption to be guaranteed. The second direction of research on the robustness measure regards the size of the disturbance that can be handled. Because the current measure is based on a linearized system, no information on the basin of attraction of the feedforward controller is obtained. In practice, the basin of attraction seems quite large, as the manual disturbances applied gave initial errors of up to 15 cm, which is 6 times the accuracy range after convergence. A measure that includes the allowable size of disturbances would give further performance specifications, which would allow for meaningful testing of the basin of attraction.

4.5 | Conclusion

In this chapter, we extended the state-of-the-art robustness analysis of feedforward controlled robots to hybrid systems. This analysis method estimates the largest parameter variation with which the cyclic trajectory of the robot is still stable. It was applied on a three DOF robotic arm and the robustness estimate indicated that the expected range of parameter uncertainties would not lead to instability. Application on a prototype resulted in position errors at the pick and place positions of less than 2.5 cm, which is small enough to perform an actual pick-and-place task. After the robotic arm was perturbed heavily, it returned to performing its task within a couple of cycles. As far as we know, this is the first time that such a task was performed by a purely feedforward controlled robotic arm.

Acknowledgement

This work is part of the research programme STW, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO).

5

The effect of the choice of feedforward controllers on the accuracy of low gain controlled robots

Michiel Plooij*, Wouter Wolfslag* and Martijn Wisse
* These authors contributed equally to this chapter,
IEEE International Conference on Intelligent Robots and Systems , 2015.

Abstract

High feedback gains cannot be used on all robots due to sensor noise, time delays or interaction with humans. The problem with low feedback gain controlled robots is that the accuracy of the task execution is potentially low. In this chapter we investigate if trajectory optimization of feedback-feedforward controlled robots improves their accuracy. For rest-to-rest motions, we find the optimal trajectory indirectly by numerically optimizing the corresponding feedforward controller for accuracy. A new performance measure called the Manipulation Sensitivity Norm (MSN) is introduced that determines the accuracy under most disturbances and modeling errors. We tested this method on a two DOF robotic arm in the horizontal plane. The results show that for all feedback gains we tested, the choice for the trajectory has a significant influence on the accuracy of the arm (viz. position errors being reduced from 2.5 cm to 0.3 cm). Moreover, to study which features of feedforward controllers cause high or low accuracy, four more feedforward controllers were tested. Results from those experiments indicate that a trajectory that is smooth or quickly approaches the goal position will be accurate.

5.1 | Introduction

High precision in robotics is usually achieved with high feedback gains. However, there are applications in which such high gains are undesirable or infeasible. For instance, in the presence of sensor noise or time delays, high feedback gains will make the robot unstable. A second example of robots in which high gains are undesirable are robots that interact with humans. In such robots, high feedback gains increase the risk of injury. These examples show that it is important to develop alternative techniques to obtain high precisions that work even on robots with limited feedback.

To that end, multiple researchers have taken the idea of feedback limitations to an extremum and have focused on executing tasks with robots without any feedback. A first example is the concept of passive dynamic walking, as introduced by McGeer [109]. Those walkers do not have motors and therefore no feedback control, and still walk with a stable gait. These gaits do not rely on the motion being stable at each point in time, rather they work due to the existence of stable cyclic motions, called limit cycles. Such cycles were later on combined with feedback control in so called limit cycle walkers [59, 66, 72]. Mombaur et al. [113, 114] found stable open loop controllers for walking and running robots by optimizing the open loop controllers for both stability of the motion and energy consumption.

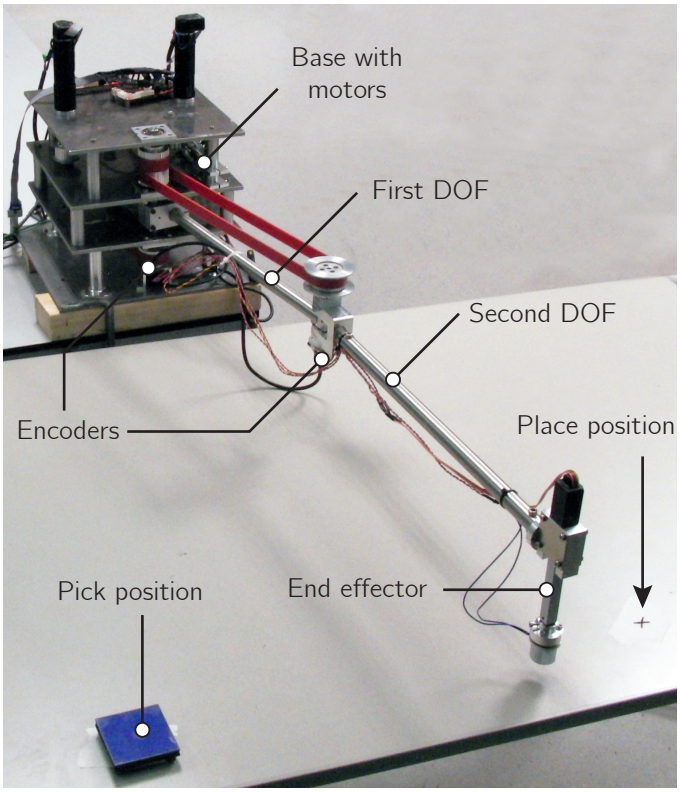


Figure 5.1: A photograph of the robotic arm we use to test our method: a two DOF SCARA type arm, which has to move between the pick and the place positions.

Control without feedback has also been applied on robotic arms. A well-known example is the work of Schaal and Atkeson [146], who studied open loop stable juggling with a robotic arm. In their case, open loop means that the state of the ball is not used as an input for the controller, but the arm itself is position controlled. In previous work, we showed that it is possible to perform open loop motions with robotic arms that are insensitive to model inaccuracies [128, 133] and to perform open loop stable cycles in which state errors vanish without any feedback [132, 185]. In [185], we optimized trajectories for open loop stability, and used an initial on-line learning approach to improve the precision of the purely feedforward controlled robot. In these studies, the trajectory itself was effected by the choice of feedforward controller, which was optimized. In the rest of this chapter, we consider the feedforward controller and the trajectory to contain the same information, since they can be translated into each other using the model of the robot.

Previous examples show that the most common technique to stabilize robots with-

out feedback is to optimize trajectories for their open loop stability [113, 114, 132, 133]. In practice, a certain amount of feedback will always be available, and therefore the advantages of both control paradigms should be exploited to achieve higher precision [91]. However, it is unclear if trajectory optimization is useful for robots with at least a little feedback.

Therefore, the question we will answer in this chapter is: *is optimizing the feedforward controller for stability a useful approach to improving the accuracy of robotic arms with (limited) feedback?* We will answer this question by studying a two degree of freedom (DOF) SCARA type robotic arm in the following way. First, in section 5.2 we explain the methods that we used, including the setup and task we study. Then, in section 5.3 we introduce the Manipulation Sensitivity Norm (MSN), which we use to estimate the lower and upper bounds of the accuracy of the arm given a certain feedback controller. Next, in section 5.4 we show the results of four alternative controllers that indicate that smooth and goal directed motions result in high accuracy. These results are discussed in section 5.5. And finally in section 5.6, we will conclude the chapter.

5.2 | Methods

In this section we explain the methods we used. First, we explain the systems under consideration, including the controller. Second, we describe the robotic arm that is used as a test case. Third, we discuss the specific task that is studied. And finally, we discuss the feedforward term in the controller.

5.2.1 System description

The type of system considered in this chapter is a serial chain robotic arm moving in the horizontal plane. The equation of motion of such a system is described by a second order differential equation:

$$\ddot{q} = f(q, \dot{q}) + M^{-1}(q)\tau \tag{5.1}$$

with q the absolute angles of the links of the robot, $\dot{q}$ and $\ddot{q}$ the angular velocities and accelerations, and τ the motor torques, which are used as control inputs. Note that this system is non-linear due to the Coriolis and centrifugal terms $f(q, \dot{q})$, and the configuration dependent mass matrix M. To control the robot, both a feedback and a feedforward term are used, hence $\tau = \tau_{\text{fb}} + \tau_{\text{ff}}$.

Because the goal of this chapter is to investigate the effect of feedback gain limitations, we structure the feedback controller in such a way that it depends on only one parameter, namely ω, which is the desired natural frequency of the controlled system. For the purpose of constructing the feedback controller from this ω, the system is simplified by neglecting $f(q, \dot{q})$, and decoupling the resulting system by

considering only the diagonal entries of the mass matrix at position $q = 0$. In other words, only considering the simplified system

$$\text{diag}(M(0))\ddot{q} = \tau \tag{5.2}$$

Note that this simplified system is only used to construct the feedback controllers and that the system we study is the non-linear system in Eq. (5.1). By using diagonal gain matrices K and C, we obtain the following, second order linear differential equations:

$$\text{diag}(M(0))\ddot{q}_i = -Kq - C\dot{q} \tag{5.3}$$

Because all matrices are diagonal, the differential equations are decoupled, which means that they can be solved separately. Finally, we choose the gains such that the natural frequency of all decoupled parts are set to a desired value (ω), and the damping ratio is set to 1, i.e. critically damped. Therefore, the controller gains are set by solving:

$$\sqrt{\frac{k}{m}} = \omega \tag{5.4}$$

$$\frac{c}{2\sqrt{km}} = 1 \tag{5.5}$$

for each part. The natural frequency ω is then used as the parameter to vary the gains. These feedback gains are then used to stabilize the robotic arm around a desired trajectory:

$$\tau_{\text{fb}} = -K(q - q_{\text{des}}) - C(\dot{q} - \dot{q}_{\text{des}}) \tag{5.6}$$

The feedforward term is simply a torque as function of time: $\tau_{\text{ff}}(t)$. This term leads to the desired trajectory: $\ddot{q}_{\text{des}} = f(q_{\text{des}}, \dot{q}_{\text{des}}) + M^{-1}(q_{\text{des}})\tau_{\text{ff}}(t)$. Because of this relation between feedforward controller and trajectory, we use the two terms interchangeably in this chapter.

The goal of this chapter is to study the effect of the feedforward controller on the accuracy of a rest-to-rest motion under disturbance. Our approach to studying the effect is to optimize the feedforward controller to minimize or maximize this disturbed accuracy. The optimization is done using single shooting, a basic optimal control approach.

5.2.2 Hardware setup

To test our approach, we use a two DOF SCARA type arm [131] (see Fig. 5.1). This type of arm was chosen as it is the simplest that can perform industrially relevant tasks. The arm consists of two 18x1.5mm stainless steel tubes, connected with two revolute joints. An end effector is connected to the end of the second tube. The motors are placed on a housing and AT3-gen III 16mm timing belts are used to transfer torques within the housing. The joints are actuated by Maxon

Table 5.1: The model parameters of the two DOF arm.

Parameter	Symbol	Value	Unit
Damping	μ_{v1}, μ_{v2}	0.2, 0.2	Nms/rad
Inertia	J_1, J_2	0.0233, 0.0312	kgm^2
Mass	m_1, m_2	0.809, 0.784	kg
Length	l_1, l_2	0.410, 0.450	m
Position of COM	l_{g1}, l_{g2}	0.070, 0.195	m
Motor constant	k_{t1}, k_{t2}	25.9, 25.9	mNm/A
Gearbox ratio	g_1, g_2	82.5:1, 22.5:1	rad/rad

60W RE30 motors with gearbox ratios of respectively 66:1 and 18:1. The timing belts provide an additional transfer ratios of 5:4 on both joints. Because the second joint is connected to its motor via a parallelogram mechanism (see [131]), the angle of the second arm is taken as the absolute angle, i.e., relative to the world frame. The end effector acts in the vertical plane and thus its motions do not influence the dynamics of the first and second DOF. The mass of the end effector is incorporated in the inertial terms of the second DOF. The arm is controlled through xPC-target in MATLAB at a frequency of 1 kHz. The parameters of this robotic arm are listed in Table 5.1.

5.2.3 Task description

We let the manipulator perform a cyclic pick and place motion, with pick and place positions at [-0.2, -0.3] rad and [0.2, 0.4] rad respectively. The time to move between the pick and the place position is 1.05 s. Hence, the total time of one cycle is 2.1 s.

5.2.4 The feedforward term in the controller

We test the accuracy resulting from different feedforward controllers: minimization and maximization of the novel Manipulation Sensitivity Norm (see section 5.3) and both smooth and time optimal trajectories (see section 5.4). The feedforward controller has to be parameterized in order to be able to optimize it.

The optimization schemes parameterize the torque signals for both joints as a piecewise linear signal, with the length of every piecewise part being 0.3 s. The space of such signals is called $\mathcal{U}$. The system states are constrained to be on the pick and place motions at the pick and place times. This also ensures that the motion is cyclic. Finally, the absolute value of both torque signals is bounded by $\tau_{\max} = 1$ Nm in order to prevent reaching the actuator limits when when feedback is needed. An example of such a feedforward control signal is shown in Fig. 5.2.

In the remainder of the chapter, the piecewise linear torque signals are optimized for various goals. These goals are expressed as a function $C(\tau_{\text{ff}})$, which is either

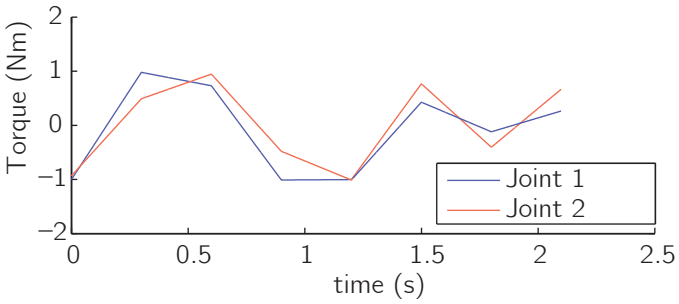


Figure 5.2: An example of a feedforward controller. The controller is parameterized as a piecewise linear function with the length of every part being 0.3 s. These torques over time are used as decision variables in the optimizations.

maximized or minimized. Combined with the task description, this leads to the following optimization problem:

$$\begin{aligned} \underset{\tau_{\text{ff}}(t)\in\mathcal{U}}{\text{minimize}} \quad & C(\tau_{\text{ff}}) \\ \text{subject to} \quad & |\tau(t)| \leq \tau_{\max} \qquad \forall t \\ & q(0) = q_{\text{pick}} \\ & q(1.05) = q_{\text{place}} \\ & q(2.1) = q_{\text{pick}} \\ & \dot{q}(0) = \dot{q}(1.05) = \dot{q}(2.1) = [0, 0] \end{aligned} \tag{5.7}$$

5.3 | Optimality study

In this section we estimate the lower bound and upper bound of the accuracy of the arm, given a certain feedback gain ω. First, we introduce a new measure for disturbance and modeling error rejection, called the Manipulation Sensitivity Norm (MSN). Then, we use this measure to optimize feedforward controllers in simulation, both minimizing and maximizing the MSN. Finally, we apply these controllers on the hardware setup, to test their accuracy.

5.3.1 The Manipulation Sensitivity Norm

To judge the quality of a feedforward signal, a measure is needed that quantifies the feasibility of performing a manipulation task when there are disturbances or modeling errors. This section explains the novel Manipulation Sensitivity Norm (MSN), which is inspired by the gait sensitivity norm used for bipedal walking robots [67]. This inspiration comes from the insight that a pick and place task can be seen as a repetitive motion and can therefore be analyzed using limit cycles

[132]. The effect of disturbances on limit cycles on bipedal robots can be captured by the gait sensitivity norm, which analyses the system based on an estimation of an input-output relation that is defined once per step. This means that the effect of a realistic disturbance profile during one step is taken as input, and a performance measure as it occurs during that step is the output. In walking robots, a possible output is the step time. A slight modification of the gait sensitivity norm can be used to analyze the performance of manipulation tasks. This modification is the MSN, and requires four steps to compute.

1. Defining output indicators
2. Defining a set of realistic disturbances as input signals
3. Obtaining the input to output relation
4. Computing the appropriate system norm of the input output relation.

The first step is to define output indicators. For pick and place tasks, output indicators are a measure of the distance of the arm to the desired path. To make the analysis as clear as possible, we use the error in the absolute angles of the links at the pick position, which is the initial position of the cycle. The MSN will compute the gain from a set of realistic disturbances to this output measure and is therefore a measure of accuracy when moving under real world disturbances.

The next step is to define the disturbances, which are used as inputs. For our analysis, we use three disturbances: a torque on the first link during the first 0.15 s of the cycle, a varying end-effector mass that represents a product that has a different weight than expected and a varying viscous friction coefficient. These inputs have a nominal value of 0, and their value is allowed to change every cycle. Note that the last two inputs, mass and friction, are typically seen as parameter variations. For this linearized analysis, there is no mathematical distinction between such a parameter variation and a more traditional disturbance such as the torque. This justifies treating parameter variations and disturbances in the same way. Do note however, that the parameter uncertainty is lasting, which should be reflected when computing the input-output gain.

In the third step, the input-output relation in Eqs. 5.8-5.9 are obtained using a finite difference scheme. At the beginning of every cycle, very small (10^{-5}) initial condition disturbances are used to obtain a Jacobian matrix A, by comparing the initial state to the state after exactly one cycle. Then, small values (10^{-5}) of the inputs are used to obtain the input to state Jacobian B. Finally, the relation to the output is linearized, again both for initial error and inputs, obtaining C and D respectively. The procedure is described in more detail in [67]. We now obtain

a state space system S of the form:

$$x(n+1) = Ax(n) + Bu(n) \tag{5.8}$$
$$y(n) = Cx(n) + Du(n) \tag{5.9}$$

where x is a vector containing the errors in the state after each cycle and y is a vector containing the errors in the positions at the pick position. In this linearized discrete system, the matrix A depends on the system dynamics and the chosen trajectory and B depends on how the inputs influence the state error. Note that for our choice, the output indicators are already linear in the state and $D = 0$.

The last step is to compute an appropriate norm, which is the main difference between the gait sensitivity norm and the MSN. For walking, it is important that the walking motion recovers to normal after the disturbance stops. For manipulation, the disturbances tend to last, meaning we are interested in the maximum error in the situation where the disturbance continues to exists, The appropriate norm is thus the induced $\mathcal{L}_\infty$ norm:

$$\|S\|_{\mathcal{L}_\infty} = \sup_{u \neq 0} \frac{\|y(u)\|_\infty}{\|u\|_\infty} \tag{5.10}$$

What remains is to compute $\|S\|_{\mathcal{L}_\infty}$. The $\mathcal{L}_\infty$-induced norm is the same as the $\mathcal{L}_1$ norm of the impulse response [20, 127]. Rather than computing the complete $\mathcal{L}_1$ norm, we approximate it by taking the sum of the first N steps, with $N = 100$, chosen sufficiently large. Take $g_{ij}(n)$ as the impulse response from input j to output i. Now the Manipulation Sensitivity Norm can be written as:

$$MSN = \|S\|_{\text{msn}} = \max_i \sum_{n=0}^{N} \sum_j |g_{ij}(n)| \tag{5.11}$$

The MSN is the amount of error given a unit input and therefore could have radians as unit. However, the specific input for which the error is largest differs per cycle. Therefore, we chose to not use a unit for the MSN. The calculation of the MSN is visualized in Fig. 5.3 and the overall procedure is summarized in Algorithm 3.

When computing the MSN, we should scale the size of the inputs in order to take into account the difference in the effect they have and the realistic sizes of those inputs. Since the expected disturbances depend heavily on the system under consideration, we choose a different approach, which highlights the capability of the MSN to take into account multiple disturbance sources at the same time. The input sizes are scaled such that the MSN of each of the three inputs considered separately is 1, when feedback gains specified by $\omega = 1\ \text{s}^{-1}$ are used for the MSN minimization.

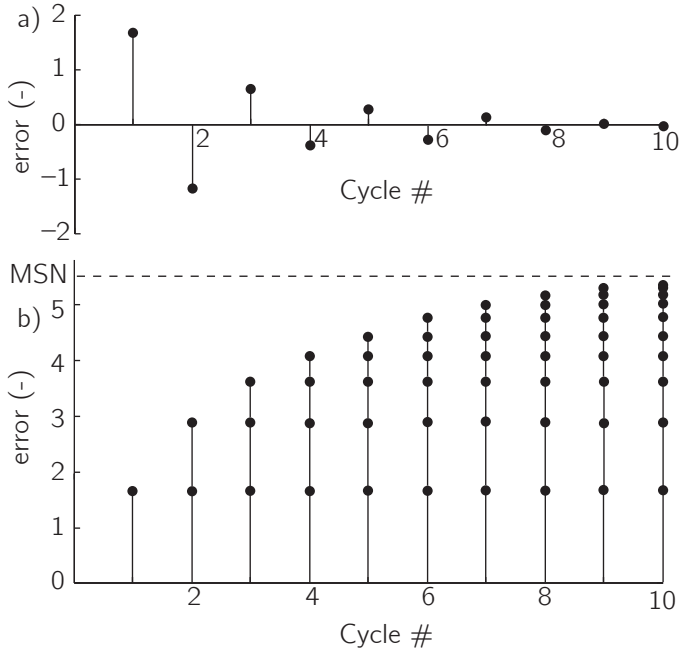


Figure 5.3: A visualization of the calculation of the MSN. a) The impulse response of the system. b) The sum of the absolute values of the impulse response. The MSN can be interpreted as the amount of error given a unit input, which would have radians as unit. Since the input for which the error is largest differs per cycle, we chose to not use a unit for the MSN.

The overall optimization for minimizing/maximizing the MSN is described by eq. (5.7), with as cost function

$$C(\tau_{\text{ff}} = \pm MSN(\tau_{\text{ff}}) \tag{5.12}$$

We used `fmincon` with 20 initial conditions determined by `multistart` in MATLAB® to solve the optimizations.

5.3.2 Simulation results

Fig. 5.4 shows the minimum and maximum MSN that were obtained as functions of the natural frequency parameter ω. This figure shows what was to be expected: increasing the gains results in a decrease of the MSN. The figure shows both the maximum and the minimum MSN that were obtained by optimization. At low gains, the maximization does not result in stable controllers, meaning that the MSN is infinitely large. This instability shows that at these gains the unstabilizing dynamical effects are larger than the stabilizing effects of the feedback controller. For $\omega > 5$ s^{-1}, the difference between the cycles with minimized and maximized MSN is negligible.

Algorithm 3 Calculating the MSN

procedure MSN($\tau(t), \omega, N$)
  Determine $[q(t), \dot{q}(t)]$ ▷ Eq. (5.1)
  Determine K ▷ Eq. (5.4)
  Determine C ▷ Eq. (5.5)
  Determine A ▷ Finite difference on q_0
  Determine B ▷ Finite difference on u
  for $j = 1..J$ **do**
    for $n = 1..N$ **do**
      Determine $g_{ij}(n)$ ▷ Eqs. 5.8-5.9
  Determine MSN ▷ Eq. 5.11
  return MSN

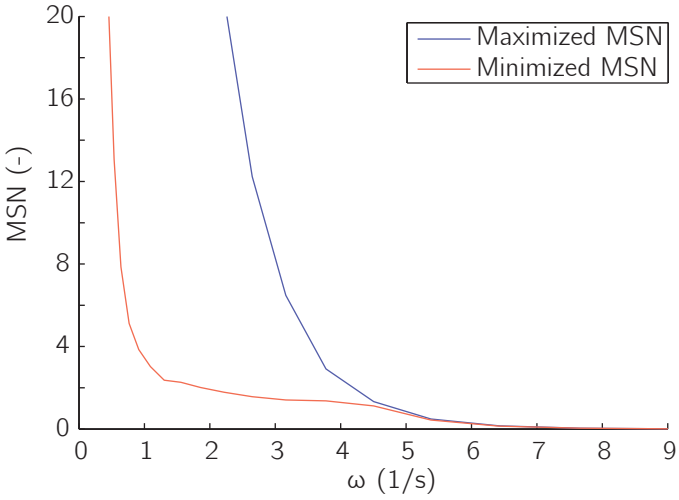


Figure 5.4: The minimized and maximized MSN as function ω, as found in simulation.

The red lines in Fig. 5.6a and 5.6b correspond to two optimized cycles. The cycle in Fig. 5.6a was obtained by maximizing the MSN and the cycle in Fig. 5.6b was obtained by minimizing the MSN. These cycles were obtained for $\omega = 2.7\ \mathrm{s}^{-1}$. The corresponding values for the MSN are 12.2 and 1.6 respectively. Finally, Fig. 5.7 shows the time evolution of the torque signals, the feedforward term of which was obtained in simulation.

5.3.3 Hardware results

Fig. 5.5 shows the position error of the end effector at the pick position during hardware experiments as function of the natural frequency parameter ω. These errors are the average error over 10 cycles after letting the robotic arm converge

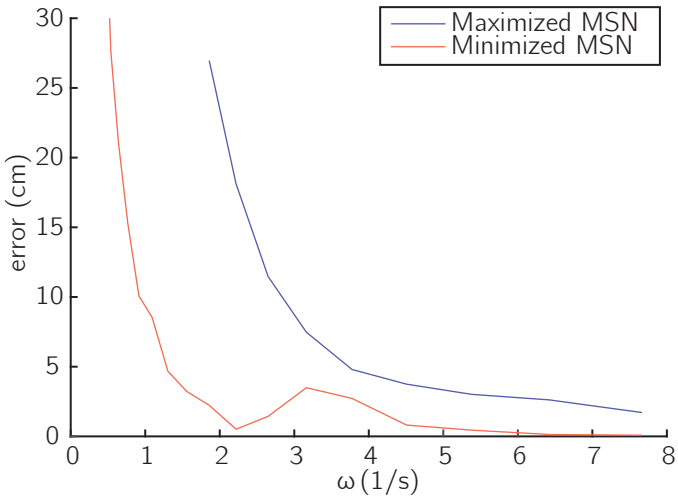


Figure 5.5: The errors at the pick position as function of ω, as found in hardware experiments. The errors are shown for trajectories with a minimized MSN and a maximized MSN.

for 2 cycles initially. The standard deviation over these 10 cycles is negligible. Logically, the errors decrease when the feedback gains are increased.

At the pick position, the error of the MSN minimizing trajectory is 0.3 cm at $\omega = 2.1$ s^{-1}, in between $\omega = 2.1$ s^{-1} and $\omega = 4.5$ s^{-1}, the error of that trajectory is approximately 2.5 cm and for $\omega >4.5$ s^{-1}, the error drops to approximately 0.3 cm again. The error of the maximized-MSN-trajectory is larger than 2.5 cm, for almost the whole range of gains. Only, for $\omega = 7.7$ s^{-1}, the error becomes 1.8 cm. This significant difference between the errors of these two trajectories means that the choice for the feedforward controller is important for the accuracy of the task execution.

Fig. 5.6 shows two typical sets of hardware results. These results were obtained for a controller with $\omega = 2.7$ s^{-1}. The plots show the state space trajectories for a maximized MSN and a minimized MSN. They show that the three trajectories differ significantly: the trajectory with maximized MSN covers a larger part of state space than with minimized MSN. The measurements on the prototype show the 10 cycles used in determining the errors used in Fig. 5.5. It can be seen that the arm has converged, and only very little variation between cycles occurs.

Fig. 5.7 shows the torques for the two different feedforward controllers with $\omega = 2.7$ s^{-1}. The plots show both the feedforward torque and the actual torque. The difference between the two is due to the feedback controller. The plots clearly show that the feedback control effort is larger when following the trajectory with maximized MSN (Fig. 5.7a), than when following the trajectory with minimized

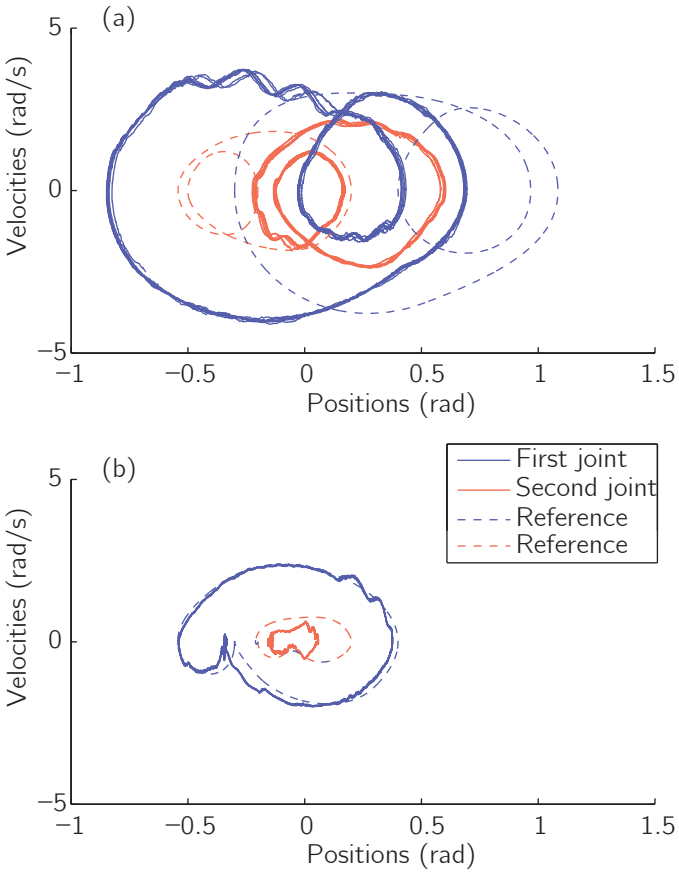


Figure 5.6: State space plots of the optimization and hardware results for ω =2.7 s^{-1}. a) The cycle with a maximized MSN. b) The cycle with a minimized MSN.

MSN (Fig. 5.7b).

5.4 | Alternative motion profiles

In the previous section, the feedforward controllers under study were determined by minimizing or maximizing the MSN. To further study which feedforward controllers lead to accurate motions, four more methods to generate a feedforward controller will now be compared. The simulation and hardware results for these controllers are found in Fig. 5.8 and 5.9 respectively.

The first of the controllers is used to compare the minimized and maximized MSN trajectories to a trajectory that is standard in industry: *a trapezoidal velocity profile*. The trapezoidal velocity profile is created by dividing the time to move

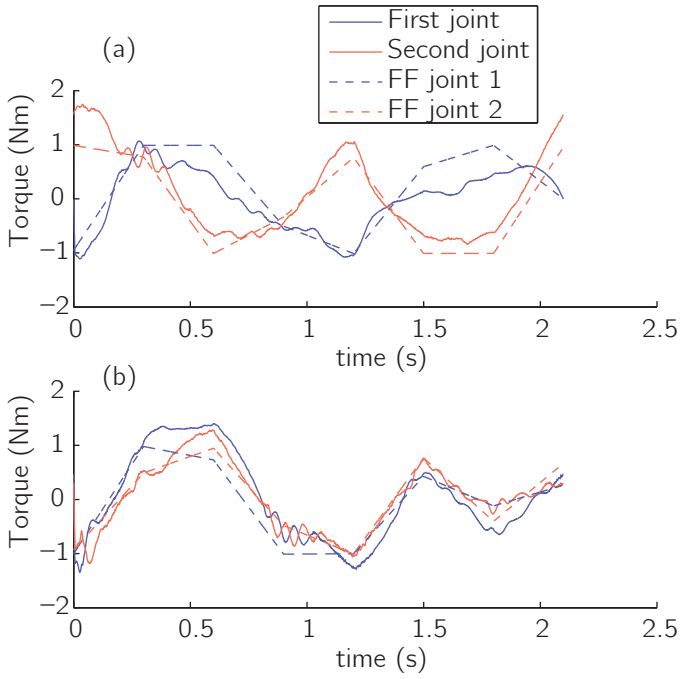


Figure 5.7: The torques as functions of time for the results with ω =2.7 s^{-1}. a) Torques for the cycle with a maximized MSN. b) Torques for the cycle with a minimized MSN.

between pick and place position in three equal parts: one part each for acceleration, constant velocity and deceleration. The same procedure is used to move from place to pick position. Both the MSN in simulation, and the position error on the hardware show that this trapezoidal trajectory has accuracy closer to the minimized MSN trajectory than to the maximized MSN trajectory. In hardware results, the error of the trajectories with minimized MSN and trapezoidal velocity profile are not even significantly different.

So, why does this standard controller perform as accurate as the optimally accurate one? There are two potentially beneficial aspects to this trapezoidal trajectory. First, it is relatively smooth, without large accelerations back and forth. Second, it approaches the goal directly, and is already close to the goal position for the last part of the motion. To test if these two effects are indeed beneficial, we compare feedforward controllers with these two specific aspects.

If smooth controllers lead to accurate motions, it should be impossible to make an inaccurate motion with a smooth controller. To test this, we performed *the MSN-maximization with a smoothness constraint*. Here we took the smoothness as a maximal torque time derivative of $t_f/4$ Nm/s. This rate allows the torque to

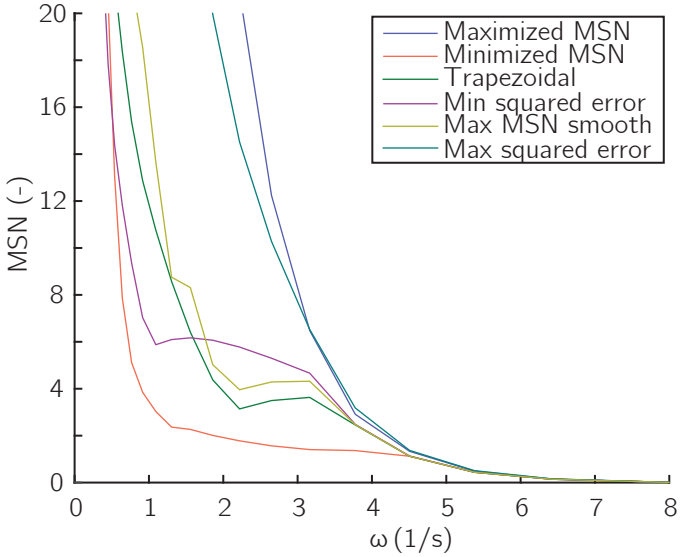


Figure 5.8: The MSN as found in simulation experiments. The errors are shown for six types of trajectories.

go from maximum to minimum and back in one cycle. As can be seen in Fig.5.9, this maximization with constraint has similar accuracy as the minimization in hardware results. This indicates that smoothness is indeed beneficial for accuracy.

To test whether a quick motion towards the goal leads to low errors, we optimized a cost function that squares the error with the goal position. This new optimization is otherwise the same, but minimizes the following cost function:

$$C(\tau_{\text{ff}}) = \int_0^{t_f} (q(\tau_{\text{ff}}) - q_g(t))^T (q(\tau_{\text{ff}}) - q_g(t)) dt \tag{5.13}$$

With $q_g(t)$ being the way-point position when $t < t_f/2$, and the initial position otherwise. Again, the results show that *this squared error minimization* gives accuracy close to the minimized MSN trajectory, as expected.

In order to confirm these results, we also tested the motion with a *maximize the squared error* function? Because this would lead to a motion that moves away from the target, and only reaches the target at the very last moment, this motion is expected to result in relatively large errors. Furthermore, moving away, and then rapidly towards the target is not very smooth, which is also likely to affect the accuracy adversely. Figs. 5.8 and 5.9 show that this prediction is indeed true. The trajectory performs worse than the other trajectories, although still not as poorly as the maximized MSN trajectory.

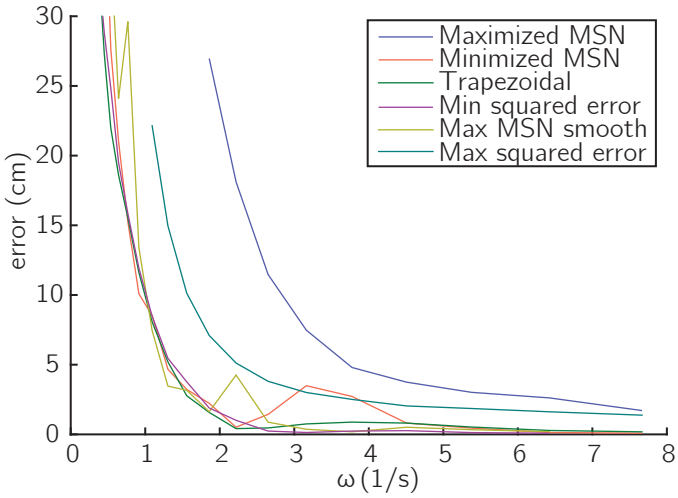


Figure 5.9: The errors at the pick position as function of ω, as found in hardware experiments. The errors are shown for six types of trajectories.

5.5 | Discussion

In this section, we discuss the results as presented in this chapter. As can especially be seen in Fig. 5.5, the choice for a certain trajectory is important for the accuracy that is achieved. For high gains ($\omega > 3.2$ s^{-1}), this choice makes the difference between a negligible position error and an error of multiple centimeters. For medium gains (1.9 s$^{-1} < \omega < 3.2$ s^{-1}), this choice makes the difference between negligible position errors and errors between 5 and 27 cm. And for low gains ($\omega < 1.9$ s^{-1}) this choice makes the difference between stability and instability.

The simulation and hardware results match well. Particularly, the shape of Figs. 5.4 and 5.5 are similar. There are however two differences. First, the difference in error between the trajectories does not converge to 0 when the gains increase in hardware experiments, whereas the difference in MSN does convergence in simulation. The second difference is the hump in error and MSN that occurs around $\omega = 3$ rad/s. These differences are small and are caused by unmodeled dynamics that were not taken into account in our choice for disturbances in the MSN-computation. The most likely effects are elasticity in the timing belts and backlash. Because the simulation and hardware results are so similar, the MSN is a good approximation for accuracy, and can be used to find feedforward controllers in cases when feedback gains are low, yet accuracy is important.

The shape of Figs. 5.8 and 5.9 are similar, but there are clear differences. The first and most important difference is that only two control strategies lead to errors that are significantly larger than the minimum error. Those two control strategies

are the maximized MSN and the maximized squared error. These results suggest that there are two principles that lead to small errors: smoothness and being close to the goal position before the end of the motion. Therefore, we expect that other common profiles such as minimal energy, minimal torque, minimal acceleration and minimal jerk will also perform well. The second difference is that in the hardware results, the accuracy of the MSN-optimized feedforward controllers only give an estimate of the range of possible accuracies. This can be seen in Fig. 5.9, in which the minimized MSN controller is not the most accurate one for $2.1\ \mathrm{s}^{-1} < \omega < 4.5\ \mathrm{s}^{-1}$.

In simulation, there are two points where one of the comparison trajectories has an MSN that is outside the range given by the MSN minimization and maximization. Specifically, this occurs for the In both these instances, the difference is small, and caused by the choice of step size in simulation. In optimization, a sampling time of 0.01 s is used to save computation time. For Fig. 5.8, a sampling time of 0.001 s was used, because this aligns with the robot hardware.

As ω goes to zero, both the MSN and the error in hardware results get very large. This is logical since the system without feedback is not stable. Fig. 5.5 shows that for feedback with $\omega < 1.8\ \mathrm{s}^{-1}$, the error is larger than 2.5 cm and therefore picking up objects of reasonable size will be difficult. If the feedback gains cannot be increased, more mechanical feedback has to be implemented. The most straightforward approach is to place springs at the joints. In our previous work [132, 185], we showed that with a spring at the first joint, tasks can be performed stably even when $\omega = 0$.

The results from this chapter can be improved by incorporating the feedback in a Repetitive Control (RC) scheme [98]. In an RC scheme, the feedforward controller is adjusted based on the state error in the previous cycle. In the most simple form, the feedforward controller in the current cycle is equal to that in the previous cycle plus the feedback that was applied. Such an RC scheme was used before on robotic arms to learn open loop stable trajectories [185].

There is an interesting parallel between the controller we use in this chapter and human movement control. Similar to our controller, humans also exploit the advantages of both feedforward and feedback in order to optimize their performance [42]. For fast motions, humans cannot rely on feedback at all, due to the large time delays (typically 150 ms for humans [36, 170]). Therefore, they have to rely on feedforward, in which control signals are generated based on the prediction of an internal model [78]. In slower motions, more feedback is used to correct for inaccuracies in the internal model and external disturbances.

Another interesting parallel with human motion control is the fact that smooth motions perform well. In human motion control, there is an ongoing debate about the cost function humans use optimize their motions. Suggested cost functions

are the maximum jerk [49, 68], change of torque [175] and sensitivity to motor noise [62]. Other researchers suggest that humans perform some kind of stochastic optimal control in which variability in task irrelevant directions is ignored [172]. The problem in this debate is that all cost functions result in approximately the same smooth motions. Similarly, we expect that all smooth motions that result from such cost functions will perform well in terms of accuracy.

5.6 | Conclusion

In this chapter we focused on the question 'is optimizing the feedforward controller for self stability a useful approach to improving the accuracy of robotic arms with (limited) feedback?' The answer to this question is: 'yes, the choice for a certain feedforward controller can make the difference between an accurate and an inaccurate task execution.' The feedforward controller was tuned using the novel Manipulation Sensitivity Norm, which measures the accuracy while taking into account disturbances and model errors. Feedforward controllers that were either minimized or maximized for this norm were implemented on our robotic arm. Results show that for a large range of feedback gains, the error varies between 0.3 and 2.5 cm, depending on the choice for the feedforward controller. Further experiments with alternative feedforward controllers indicated that a trajectory that is either smooth, or approaches the goal position quickly, will be accurate. Therefore, the commonly used trajectory with a trapezoidal velocity profile performs well and is a good choice in terms of accuracy.

Acknowledgement

This work is part of the research programme STW, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO).

6

Dissipatively actuated manipulation

W.J. Wolfslag, M.C. Plooij, W. Caarls, S. van Weperen and G.A.D. Lopes,
Control Engineering Practice, Vol 34, 2015.

Abstract

This chapter addresses the design of control systems whose actuation can only dissipate energy. Such systems provide intrinsic safety, and can be used in scenarios where energy is supplied by external entities and point-stabilization is possible with only energy dissipation. Three control synthesis methods are proposed that range from model-based to a learning approach and their validity is demonstrated on a passively controlled manipulator performing a positioning task. These three methods are the Zero Control Velocity Field, Monte-Carlo Tree Search and Reinforcement Learning. The simulation results are corroborated by experiments on a physical two link manipulator.

6.1 | Introduction

On many robotic arms, actuators actively provide the power needed to achieve motion. Designers often aim for the cheapest or lightest actuator that suffices the power requirements, in order to reduce the cost and weight of the arm. This chapter takes this aim to an extremum and proposes control systems for a manipulator that does not use active actuators to power the motion. In such a manipulator, motions can only be influenced by clutches and dissipative components such as brakes. As a case study for this idea, the system shown in Figure 6.1a is examined: a two DOF manipulator in the vertical plane which picks up objects and places them at a lower height. While doing so, the controller can dissipate energy but cannot add energy to the manipulator. Such a system shows resemblance with a skier (see Figure 6.1b), who can steer and brake by dissipating energy while going down to end up at a desired location.

Systems with solely dissipative components have several advantages. First, they exhibit intrinsic safety: such systems cannot make unexpected motions caused by unexpected inputs. Therefore, the motions are very transparent for the user, causing the system to be safer. The advantage of intrinsic safety is that it does not rely on active control, as in [38], or clever trajectories enforcing safety during control failures, as in [69], both of which can potentially fail in operation. Obtaining intrinsic safety was the prime motivation for previous research on dissipatively actuated systems in the field of haptic devices [108, 143, 165], particularly for rehabilitation purposes[6, 41]. Second, manipulators without motors are cheaper in both purchase (actuators are expensive) and usage (lower energy costs). The energy considerations prompted research in walking robots to consider purely unactuated robots [109], and derived robots which use very limited actuation combined with the dynamic properties of walking [59, 66, 72]. Finally, dissipatively actuated manipulators could lead to lightweight designs.

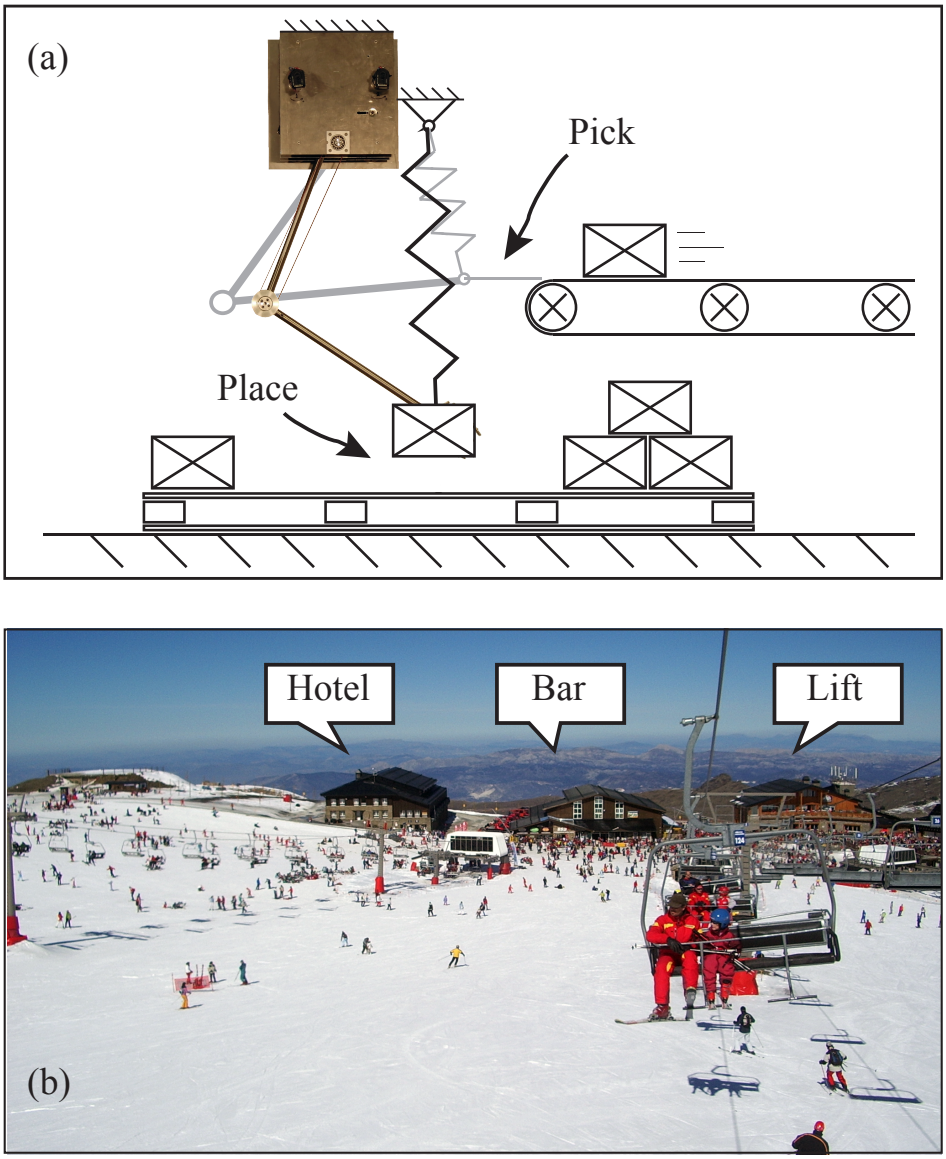


Figure 6.1: This chapter considers the control of systems with dissipative components instead of actuators. A two DOF arm in the vertical plane is studied, which picks up objects and places them at a lower height (a). While moving downwards, the arm can only brake. This case study shows resemblance with a skier that can only brake while going downslope (b), in order do steer itself to a certain goal at the bottom of the hill (e.g. a hotel, a bar or the entrance of the lift).

The goal of this chapter is to introduce the concept of *dissipatively actuated manipulation*. This concept turns out to be challenging from a controls perspective,

as the resulting systems do not fulfill the required assumptions of the traditional control approaches for nonlinear input-affine models. Therefore we compare three controllers that do not make these assumptions: a Zero Control Velocity Field (ZCVF) controller, a receding horizon controller implemented by Monte-Carlo Tree Search (MCTS) and a Reinforcement Learning (RL) controller. These three controllers merely illustrate a much wider range of possible controllers. They are chosen as they form a scale from a mostly model-based towards a purely numerical approach. As such, they represent three main paradigms in control: model-based, receding horizon and learning.

Past research into the control of dissipatively actuated systems has largely been guided by the requirements of haptic devices. Two prime topics of interest are the regulation of admittance using dissipative actuators [60, 61], and the creation of position dependent steering forces at an end-effector [6, 54]. In both these cases the desired behaviour is expressed locally, which is not possible in our tasks: position control.

More global behaviour was obtained in research on walking-guidance by means of wheeled robots [83]. However, as the desired behavior of that robot was guidance, the control system was based on a user supplying a force in approximately the right direction. Furthermore, the main external force, gravity, was compensated for using active actuators. For these two reasons their approach is not directly applicable to the problem here. The zero-control-velocity-field controller (discussed in section 6.4), is the most similar to their approach, as it tries to steer towards a preplanned trajectory.

Outside of haptic guidance, dissipative actuators are less commonly used. In underactuated robot manipulators, brakes have been used as a locking mechanism, allowing the robot to reach a desired configuration by sequentially manipulating and locking the unactuated degrees of freedom [5, 30, 119]. Although such a decoupling mode could be used in a fully dissipatively actuated robot arm, the decoupling structure severely limits the possibility of control.

The rest of this chapter is structured as follows. Section 6.2 formulates the dissipatively actuated manipulation problem mathematically. Section 6.3 describes the test case chosen to test the controllers: a two DOF manipulator in the vertical plane. Sections 6.4, 6.5 and 6.6 discuss the three controllers compared in this chapter: ZCVF control, MCTS control and RL control. The chapter ends with a discussion in Section 6.7 including a comparison between the three controllers and a conclusion in Section 6.8.

6.2 | Dissipatively actuated systems

Consider a class of mechanical systems in which the controller cannot add energy to the system. For mechanical systems, the energy is expressed as [23]:

$$H(q,p) = \frac{1}{2}p^T M(q)^{-1} p + V(q) \tag{6.1}$$

where $q \in \mathbb{R}^n$ the generalized coordinates, $p = M(q)\dot{q}$ the generalized momenta, $M(q)$ the positive definite mass matrix, and $V(q)$ the potential energy.

The equations of motion can now be expressed in port-Hamiltonian form [147]:

$$\begin{aligned} \begin{pmatrix} \dot{q} \\ \dot{p} \end{pmatrix} &= \begin{pmatrix} 0 & I \\ -I & -R \end{pmatrix} \begin{pmatrix} \nabla_q H \\ \nabla_p H \end{pmatrix} + \begin{pmatrix} 0 \\ I \end{pmatrix} u \\ \xi &= \begin{pmatrix} 0 & I \end{pmatrix} \begin{pmatrix} \nabla_q H \\ \nabla_p H \end{pmatrix} \end{aligned} \tag{6.2}$$

where $u \in \mathbb{R}^n$ are the input torques, R a positive definite matrix representing mechanical damping (e.g. friction), and $\xi = \nabla_p H = \dot{q}$, is the output of the system.

From this formulation, one can show that the system is *passive* with the total energy H as storage function, as can be seen by expressing the time derivative of H:

$$\dot{H} = -\dot{q}^T R \dot{q} + \xi^T u \leq \xi^T u \tag{6.3}$$

The input does not add energy to the system as long as $\xi^T u \leq 0$, or equivalently:

$$\dot{H} \leq -\dot{q}^T R \dot{q} \tag{6.4}$$

In this case, the controlled system is passive with respect to the original Hamiltonian, even if there is no damping ($R = 0$). Denote systems (respectively controllers) that satisfy eq. (6.4) as *dissipatively actuated systems* (respectively dissipatively actuated controllers). Such systems should be distinguished from dissipative control systems, described in for instance [79],[120], in which both the controller and the uncontrolled system meet a demand similar to Eq. 6.3. Do note that a dissipatively actuated system is always a dissipative control system, the converse is not true.

A more strict demand is

$$\xi_i^T u_i \leq 0 \qquad \forall i = 1, \ldots, n \tag{6.5}$$

which gives rise to *elementwise dissipatively actuated systems*. This distinction is important in implementation, because elementwise dissipatively actuated systems

can be constructed using only brakes, whereas dissipatively actuated systems in general require clutches. The implementations discussed later in this chapter are all elementwise dissipatively actuated controllers.

The control goal is dissipatively actuated manipulation: steering a dissipatively actuated system to a desired state. The first challenging aspect of dissipatively actuated manipulation can be understood from eq. (6.4). Since the energy in the system must always be decreasing, the problem of point stabilization becomes challenging: if the energy of the desired final state is higher then the total energy of the initial condition, then there exists no solution.

Even neglecting such reachability issues, the design of dissipatively actuated controllers is not solved by standard methods. To see this, consider the following structure of an elementwise dissipatively actuated controller:

$$u = -D(q, \dot{q})\dot{q} \tag{6.6}$$

where D is positive diagonal. If instead of this diagonal constraint, $D + D^T$ is constrained to be positive definite, a normal dissipatively actuated system is obtained, as Eq.6.4 is then satisfied. In either case, if D is allowed to be discontinuous, all (elementwise) dissipatively actuated state feedback controllers can be written in this form.

The constraints on D have two important consequences for model based controller design: First, they mean the system is no longer small-time locally controllable [23], as can be seen from the reachability argument above. Secondly, while the system is input affine, its control space is not proper, i.e. 0 is on the boundary of the control space. Unfortunately, most results on input-affine systems (see [23][79]) require the input space to be proper, or even to be $\mathbb{R}^n$. As such, the requirements for most nonlinear model-based control synthesis techniques (e.g. feedback linearization, backstepping or sliding mode control; see [79]) are not met. Furthermore, when using a flatness based approach [176], the constrained space is not readily expressed in bounds on the flat output and its derivatives. This makes a flatness based approach unappealing as well.

It is concluded that the problem of dissipatively actuated control is not readily solved by standard model-based control design methods. In the remainder of the chapter three approaches are discussed that have a significant degree of numerical planning contained in them (this is also the case for other types of more challenging plants such as nonholonomic systems).

6.3 | Case study

To test the three proposed approaches, a pick and place task for a two DOF arm in the vertical plane is used. Before discussing these approaches, the hardware

setup is described, how it is modeled and the task it has to perform.

6.3.1 Robotic manipulator

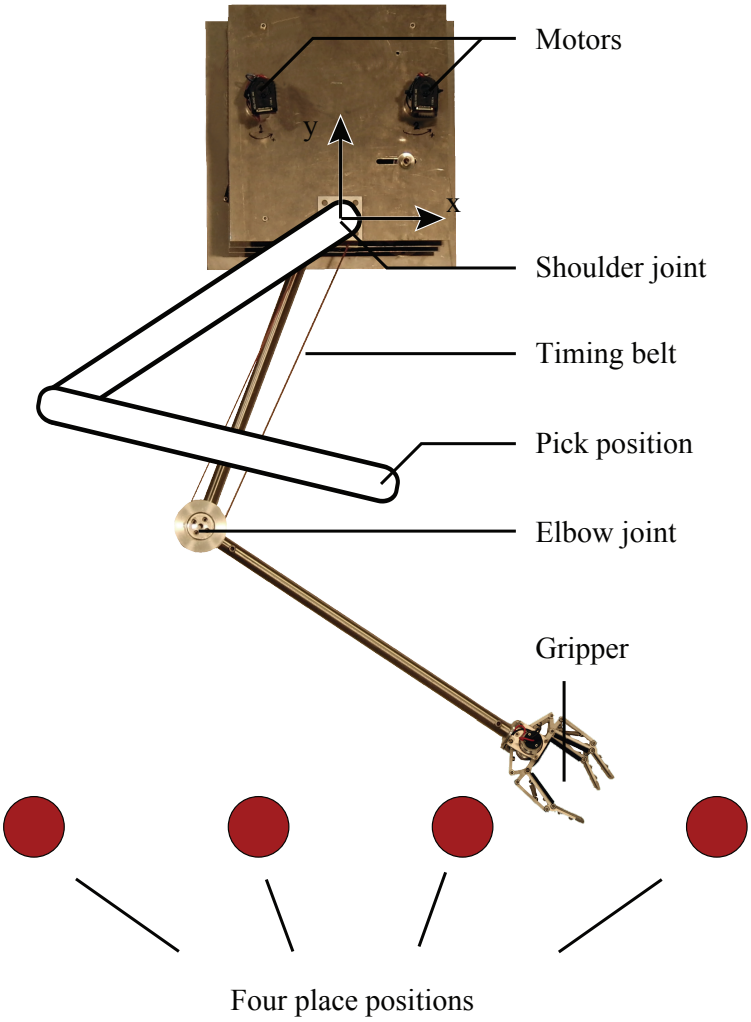


Figure 6.2: This figure shows the task of the case study considered in this chapter. The two DOF arm has to move from an initial position (pick position) to four different goal positions (place positions), that are at a lower height than the initial position.

Figure 6.2 shows a picture of the two DOF robotic arm [131]. The DOFs are created by two 18x1.5mm stainless steel tubes, connected with two revolute joints. A mass of 1 kg is connected to the end of the second tube, which represents the weight of a gripper plus a product. The motors are placed on a housing and AT3-

Table 6.1: The model parameters of the two DOF arm.

Parameter	Symbol	Value	Unit
Coulomb friction	μ_{c1}, μ_{c2}	0.25, 0.08	Nm
Viscous friction	μ_{v1}, μ_{v2}	-0.024, -0.009	Nms/rad
TD-friction	μ_{t1}, μ_{t2}	32, 22	%
Inertia	J_1, J_2	0.0364, 0.0808	kgm^2
Mass	m_1, m_2	0.8, 1.0	kg
Length	l_1, l_2	0.38, 0.55	m
Position of COM	l_{g1}, l_{g2}	0.07, 0.25	m
Motor constant	k_{t1}, k_{t2}	25.9, 25.9	mNm/A
Gearbox ratio	n_1, n_2	1:22.5, 1:82.5	rad/rad

gen III 16mm timing belts are used to transfer torques within the housing. The joints are actuated by Maxon 60W RE30 motors with gearbox ratios 1:18 and 1:66 respectively. The timing belts provide an additional transfer ratio of 5:4 on both joints. The parameters of this robotic arm are listed in Table 6.1.

The friction in the joints is modeled as a combination of viscous, Coulomb and torque dependent friction (TD friction in Table 6.1):

$$T_f = -\mu_v \cdot \omega - (\mu_c + \mu_t \cdot |u|) \cdot \text{sign}(\omega) \quad \text{for} \quad \omega \neq 0 \tag{6.7}$$

$$T_f = -(\mu_c + \mu_t \cdot |u|) \cdot \text{sign}(u) \quad \text{for} \quad \omega = 0 \tag{6.8}$$

The standard mechanical equations are derived from the port-Hamiltonian model presented in (6.2), using the Legendre transformation, to obtain:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + \nabla_q V = u + T_f \tag{6.9}$$

where M is the positive definite mass matrix, C the matrix representing the Coriolis and centrifugal forces, and V the potential energy due to gravity. Further details of these equations are omitted from this chapter due to length. Note that because of the parallel mechanism created by the timing belt to the second joint, the angle of the second arm is taken as the absolute angle, i.e., relative to the world frame.

6.3.2 Pick and place task

The manipulator has to perform a pick and place task with one pick position and four place positions (see Figure 6.2). Every motion starts at the pick position while knowing the place position it has to go to. When the place position is reached, the arm is transferred back to the pick position. In the sketch of the idea in Figure 6.1, this transfer is done by a spring once the package is dropped and the total mass

of the manipulator reduces. The actual robot setup used in this chapter does not include such a spring and thus the arm is moved up using the motors.

For mechanical systems, disturbance handling is often an important issue. In the design of the three controllers, these issues are not explicitly addressed, as the focus is on the challenge of controlling a robot arm to a desired position using only dissipative actuators. The capability of handling large disturbances is discussed in Section 6.7 as part of the reachability question, as it relates directly to the range of motions that the robot can perform.

6.4 | Zero Control Velocity Field

6.4.1 Steer towards minimal control

The first approach proposed is a model based controler, based on a simple heuristic: when control authority is limited, steer towards a trajectory that requires little or no control, and then keep following that trajectory using the natural dynamics. The idea behind this heuristic is that if a disturbance would act upon the system in a trajectory that uses no (or very limited) control, it is likely that enough control authority remains to steer back to the desired trajectory. Contrast this to the case where all control authority is required to follow the trajectory, which is likely to make it impossible to recover from a disturbance.

The above heuristic is applied using a three staged approach. The first stage is to find trajectories towards the goal position that use a control action that does not come close to violating the constraint in eq. (6.5). The second stage is defining a velocity field for which the previously found trajectories are attractive. The third stage is the design of a controller that tracks the velocity field. This final controller is then used during the motion. In the description below it is shown how these three stages are incorporated into the controller for the vertical pick and place task. The complete control procedure is summarized in Algorithm 4.

Note that this approach is not a unique approach to designing a model based controller for passively actuated systems. The design heuristic, and its implementation in the form of a velocity field controller were chosen due to their intuitive appeal. Splitting the control design in these three steps allows reasoning about the capacity of a passively actuated controller to execute the desired control commands.

There is a class of controllers, known as intrinsically passive controllers [161], which also seem to have intuitive appeal. In this case, the appeal comes from the tight link between the control design and the energy in the system, which is highly important for dissipatively actuated systems. The prime appeal of intrinsically passive controllers is that they guarentee passivity of the whole system, even with

Table 6.2: ZCVF control parameters.

Goal x-position [m]	Vertical velocity [m/s]	Virtual damping [Nms/rad]
0.3	-0.3	2.0
0.1	-0.2	2.4
-0.1	-0.4	2.0
-0.3	-0.5	1.4

model uncertainties. However, the passivity of a system that is passively actuated is already guaranteed, as discussed in the previous section. Furthermore, it is unclear how to design and connect an intrinsically passive controller with internal energy, such that eq. (6.5) is always satisfied.

6.4.2 Implementation

Stage 1, trajectory

To find the minimum control trajectories backward simulation is used. A trajectory $q_{\text{traj}}(t)$ is computed by starting at the goal position and simulating backwards in time, using a feedforward signal (i.e. a constant gain damping coefficient). This trajectory is represented by letting the workspace x-position be a function of the workspace y-position, resulting in the trajectory always going downwards at all time. This also makes the trajectory not cross itself.

In the backward simulation, there are two parameters that have to be tuned. The first tunable parameter is the velocity at the goal position. In a practical application, this goal velocity is typically 0 rad/s. However, a large braking torque towards the end of the motion can easily be used to achieve that velocity. Therefore any goal velocity can be used to tune the motion in the backwards simulation[1]. A final braking step is then used to obtain the actual goal velocity on the hardware. This braking step will clearly dissipate energy, so a standard controller can be used without requiring an actuator that provides energy. Because such a standard controller can be used, this braking step is ignored in the rest of this chapter. The second parameter to tune for the backward simulation is the virtual damping coefficient. A diagonal matrix is used to specify the simulated damping, where the damping coefficients are the same for each link, leaving just one tunable parameter. The velocity and damping are manually tuned to obtain trajectories with a low endpoint velocity that do not require large braking torques. Table 6.2 shows the damping and velocities used for the goal positions in the experiment.

[1] If the final velocity given by the control specification is nonzero, a similar reasoning applies. The only change is that the final velocity for the simulation must be larger than the required velocity.

Stage 2, velocity field

A velocity field $\dot{q}_{\text{des}} = N(q)$ is created such that $N(q_{\text{traj}}(t)) = \dot{q}_{\text{traj}}(t)$, implying that q_{traj} is an invariant set for the velocity field. Furthermore, this invariant set is made to be attractive, that is: one must construct a Lyapunov function $\mu(q)$, which has its minimum on all points of the trajectory, such that $\dot{\mu} = \nabla_q \mu(q) N(q) \leq 0$. The vector field $N(q)$ is designed by constructing a velocity field in the end-effector coordinates (x, y), and transforming that velocity field back to configuration space coordinates, see Figure 6.3. First the uncontrolled trajectory is parameterized. This is achieved by using q_{traj} to parametrize the x-position of the endpoint trajectory as a polynomial of its y-position: $x(y) = p_x(y)$, and then have an error function $\bar{G}(x, y) = x - p_x(y)$. The velocity magnitude is also needed, which is also taken as a polynomial in y: $\sqrt{\dot{x}^2 + \dot{y}^2} = p_v(y)$. Fitting of both these polynomials is done using least squares approximation.

The next step is to define two vector fields $\bar{f}_{\text{d1}}$ and $\bar{f}_{\text{d2}}$. The first, $\bar{f}_{\text{d1}}$, points towards the desired trajectory q_{traj}. The second, $\bar{f}_{\text{d2}}$, enforces the correct velocity once the system is on the trajectory q_{traj}. A simple way to define these vector fields is as follows:

$$\bar{f}_{\text{d1}} = -p_v(y)\text{normalize}(\nabla(\bar{G}(x, y)^2)) \tag{6.10}$$

$$\bar{f}_{\text{d2}} = p_v(y)\text{normalize}\begin{pmatrix} \nabla_y(p_x(y)) \\ 1 \end{pmatrix} \tag{6.11}$$

The complete vector field $f_{\text{d}}(x, y)$ is a weighted combination of the two fields mentioned above. When x is close to $p_x(y)$, the vector field steers mostly along the trajectory. Oppositely, when x is far from $p_x(y)$, steer happens towards the trajectory. This intuitive approach is implemented using the weighting function $n(x) = (1 + \bar{G}(x)^2)^{-\alpha}$, with $\alpha \in \mathbb{R}^+$ a parameter that manipulates the trade-off between how quickly the field steers towards the desired trajectory, and how much acceleration is need to follow the vector field. For the experiments presented in this chapter $\alpha = 100$. The resulting vector field is:

$$f_{\text{d}}(x, y) = (1 - n(x))\bar{f}_{\text{d1}} + n(x)\bar{f}_{\text{d2}} \tag{6.12}$$

This field is defined in workspace coordinates, and is converted to configuration space by means of the Jacobian of the workspace-configuration space transformation to obtain $N(q)$.

Stage 3, field controller

To control towards the velocity field, an inverse dynamics approach is used [85]. Let $\bar{N}(q, \dot{q}) = \dot{q} - N(q)$, be the velocity error. Then the input u is determined by:

$$\begin{aligned} u = -K\bar{N}(q, \dot{q}) - \nabla_q \mu(q) + M(q)\nabla_q N(q)\dot{q} + \\ + C(q, \dot{q})N(q) + B(q, \dot{q}) \end{aligned} \tag{6.13}$$

where, $K \in \mathbb{R}^{n \times n}$ is a linear feedback-gain, and can be set to any positive definite matrix. For the experiments only the diagonal terms are used, which were tuned manually to obtain: $K = \text{diag}(30, 30)$. Also, $\mu(q) = \bar{G}(x, y)^2$ is a valid Lyapunov function, satisfying $\nabla_q \mu(q) N(q) \leq 0$. And finally, $B(q, \dot{q}) = \nabla_q V(q) - T_{\text{f}}(q, \dot{q})$, is a term compensating for gravity and friction. Using the candidate Lyapunov function

$$L = \frac{1}{2} \bar{N}(q, \dot{q})^T M(q) \bar{N}(q, \dot{q}) + \mu(q) \tag{6.14}$$

one can show [85] that the input u in equation (6.13) renders system (6.9) stable at $\bar{N}(q, \dot{q}) = 0$. Note that the control law described above is not necessarily a dissipatively actuated controller. Therefore u is projected on the allowed control space, using unweighed least squares. Finally, because of hardware limitations a torque saturation of 3.5 Nm is used on the first motor and 2.5 Nm on the second motor.

Algorithm 4 Zero-Control Velocity field

```
function ZCVF
    perform backwards simulation to obtain q_traj
    approximate resulting trajectory as a polynomial p_x(y)
    compute velocity field N(q) based on approximate trajectory with Eqs. (6.11)-(6.12)
    measure state [q, q̇]^T
    while floor not reached do
        find desired input torque u with velocity field and Eq. (6.13)
        if torque violates (elementwise) dissipatively actuated constraint then
            project input torque on (elementwise) dissipatively actuated inputs
        apply braking torque u
        measure state [q, q̇]^T
```

6.4.3 Results

The velocity field for moving towards $x = -0.3$ m is shown in Figure 6.3. In this figure, the blue line shows the backwards integrated motion, the red line shows the polynomial approximation of that motion, and the black arrows show the velocity references for the specific positions. Four such velocity fields were obtained, one for each goal position.

Figure 6.4.a-top shows the four resulting motions of the arm in simulation. The figure shows that the arm moves toward the four goal positions, but there is an error in the position at which the arm reaches the ground. The average position error for the four positions is 1.9 ($\pm$1.6) % of the length of the arm. Because the

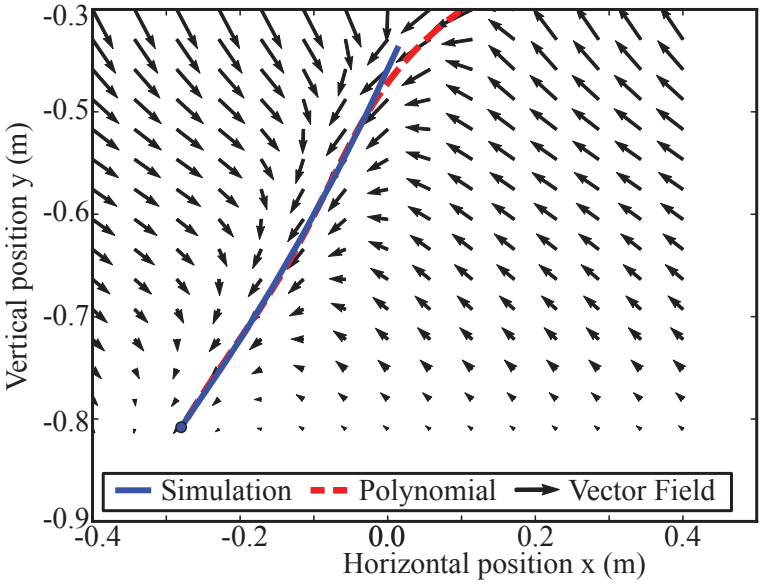


Figure 6.3: The velocity field for a controller that steers towards the most left goal position. The goal position is indicated by a circle at the height of the floor. The blue line shows the backwards integrated motion from the goal position, the red line shows the polynomial approximation of this motion, and the black arrows indicate the velocity field.

arm is nearly 1 m long, this percentage is very close to the absolute error measured in cm.

Figure 6.4.a-bottom shows the four resulting motions for the same controller, but now implemented on the robotic arm. The motion to x =0.1m and x =0.3m have the largest final errors (4.3 % and 2.3 % respectively). The average position error of the four positions is 1.9 ($\pm$ 1.6) %, which is the same as in simulation. Looking closer at Figures 6.4.a), one can see that not only the endpoints are the same, but that the whole motion is quite similar for both simulation and experiment. This similarity is likely due to a reasonably accurate model combined with high gains.

6.5 | Monte-Carlo Tree Search

The second proposed approach uses receding horizon control to steer the system in the right direction. This means that at every time step the model is used to plan from the current state to the goal, but only the first action in the plan is taken. This should reduce the requirements on the accuracy of the model because the error is continually corrected on-line.

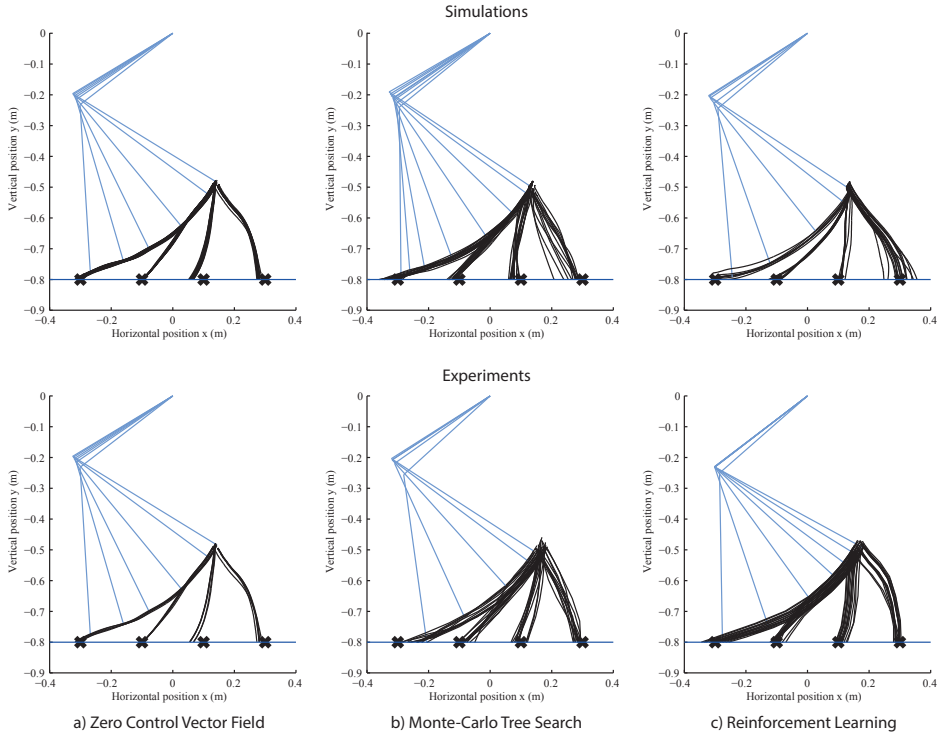


Figure 6.4: Simulation and experimental results using the a) zero control velocity fields, b) Monte-Carlo search tree, and c) reinforcement learning. The black lines represent the trajectories towards the four goal positions, which are indicated by black crosses. The blue lines show different positions of the two DOF arm, while moving towards the most left goal position.

6.5.1 Controller

A standard Monte-Carlo Tree Search (MCTS, [22]) method is used, summarized in Algorithm 5. The tree consists of nodes ν associated with a state $s(\nu)$, which is defined as $s = [q, \dot{q}]^T$. These nodes each have a number of children equal to the number of actions possible in that state. First, a root node ν_0 is created at the current state s_0. Then, while there is still time, a node ν_t is iteratively selected from the tree, perform a *rollout* from that state $s(\nu_t)$, and back up the resulting cost Δ through the tree. When time is up, the action with the lowest cost from the root node is returned.

Node selection is done by traversing the tree using an ϵ-greedy policy, selecting a random child with probability ϵ and the child that has the lowest cost otherwise. When a node is reached for which not all actions have been evaluated yet, a random unevaluated action u is expanded and the resulting child node ν_t is returned. Rollouts are performed by taking random actions starting at $s(\nu_t)$ until the arm

Algorithm 5 Monte-Carlo Tree Search

```
function MCTS(s_0)
    create root node ν_0 with state s_0
    while within computational budget do
        ν_t ← SELECT(ν_0)
        Δ ← ROLLOUT(s(ν_t))
        BACKUP(ν_t, Δ)
    return u(BESTCHILD(ν_0))
```

hits the ground or until the evaluation horizon is reached. The total cost Δ of this rollout is assigned to ν_t. Then, for all parent nodes ν up until the root node, the cost $c(\nu)$ is set to

$$\frac{k(\nu)c(\nu) + \Delta}{k(\nu) + 1}, \tag{6.15}$$

where $k(\nu)$ is the number of times ν has been selected before.

Dissipativity of the control actions is ensured by parametrizing the actions such that they always meet the dissipativity constraint. This can always be done by projecting the action on the feasible actions. For the elementwise case a more elegant approach is to only use actions with a positive value, and multiply the action-value with the sign of the velocity and the torque limit (3.5 and 2.5 Nm for the first and second joint respectively). That approach was used in the experiment, with actions of 0, 0.5 or 1.

A control step time Δt of 50 ms is used, together with a horizon of 20 steps, exploration probability 0.05, and the following cost function:

$$c(s) = \begin{cases} -e^{-5|\mathrm{x}(s)-\mathrm{x}(s_{\mathrm{des}})|} & \text{if } \mathrm{y}(s) < -0.8 \\ 0 & \text{otherwise} \end{cases} \tag{6.16}$$

6.5.2 Results

The performance of MCTS greatly depends on the number of rollouts that can be performed before the time budget is exhausted. We have observed that a single core of an Intel Xeon E5-2665 is able to perform ∼600 simulation steps per control step. This resulted in an average error of 2.8% of the arm length, mostly due to the large error for the far left target (see Figure 6.4.b-top). The error is mostly due to the limited number of simulations, as a warm start variant (which initializes the search using the chosen branch of the tree generated in the previous control step) resulted in an average error of just 0.1%. However, model discrepancies prohibit the use of such an algorithm on the real system.

When executing the controller on the robot (Figure 6.4.b-bottom) the error is similar to that in simulation, being 3.0 %. However, instead of falling short on the far left target the system now overshoots it.

6.6 | Reinforcement Learning

The previous two model-based controllers are compared to a completely model-free controller learned through temporal difference (TD) learning [164]. In a period before the actual task is to be performed, the learning algorithm interacts with the system to optimize the controller. Through trial and error it learns a feedback controller that minimizes the cost without the direct or indirect use of a model.

6.6.1 Controller

In model-free TD learning, one tries to estimate a state-action value function Q^π which indicates for each state-action pair (s, u) the expected cost-to-go of taking action u in state s and following the control policy $\pi : s \to u$ afterwards. Because in this case only the final error is being optimized, the cost-to-go is simply the cost of the final state. In TD control the policy π is derived from the value function by choosing the action with the lowest estimated cost-to-go. We may use the same ϵ-greedy policy as in MCTS, such that

$$\pi(s; Q) = \begin{cases} \min_u Q^\pi(s, u) & \text{with probability } 1 - \epsilon \\ \text{random} & \text{otherwise} \end{cases} \tag{6.17}$$

The optimal value function Q^*, defining the optimal policy π^*, is the unique solution to the Bellman equation

$$Q^*(s, u) = c(s') + \min_{u'} Q^*(s', u') \tag{6.18}$$

where s' is the next state when taking action u. Q is estimated by sampling state transitions $(s, u) \to (c(s'), s')$ along a trajectory and updating Q towards minimizing the temporal difference error δ. We use the SARSA algorithm, summarized in Algorithm 6.

In this algorithm, α is a learning rate that sets the parameter of an exponential moving average filter determined by the stochasticity of the problem. To speed up convergence the SARSA(λ) algorithm is used, which not only updates the current state but also a number of previous states based on their *eligibility* [164].

The same cost function as in MCTS is used, and just as in MCTS the dissipativity of the control actions is enforced by using positive actions which are multiplied with the sign of the velocity. α and λ are set to 0.1 and 0.92, respectively.

Algorithm 6 On-policy TD control (SARSA)

```
1: function SARSA
2:     obtain initial value function Q
3:     while not converged do
4:         move robot to start position
5:         obtain state s and set u ← π(s; Q)
6:         while floor not reached do
7:             apply braking torque u.
8:             obtain new state s′ and set u′ ← π(s′; Q)
9:             δ ← c(s′) + Q(s′, u′) − Q(s, u)
10:            Q(s, u) ← Q(s, u) + αδ
11:            s ← s′, u ← u′
12:    return Q
```

6.6.2 Results

In general, reinforcement learning can take quite some time before converging to an acceptable policy. However, the informative reward structure and episodic nature of the task allow for fast convergence in this case. In experiments presented here, learning took ∼2 minutes of interaction time (see Figure 6.5) with an average error of 1.2 % of the length of the arm in simulation and 1.6 % on the real setup. However, in both cases there was considerable variance between trials that learned to move to the same position.

The resulting trajectories (Figures 6.4.c) show a tendency for the simulated controller to arrive more vertically, although the observed variance prohibits firm conclusions.

6.7 | Discussion

6.7.1 Comparison

The simulation and experiment results obtained show that the three different approaches to the dissipatively actuated control problem show promising results. The question now becomes: how do these approaches compare on the general case. To answer this question one must explicitly steer away from detailed quantitative analysis of the results on this test-problem, with these implementations, as such details are unlikely to transfer to other problems.

To compare the controllers qualitatively, five criteria are used: the required accuracy of the model, the planning/learning computational cost, the online computational cost, the accuracy of motion and the range of motion that the controller allows (reachability). The results of the comparison are summarized in table 6.3.

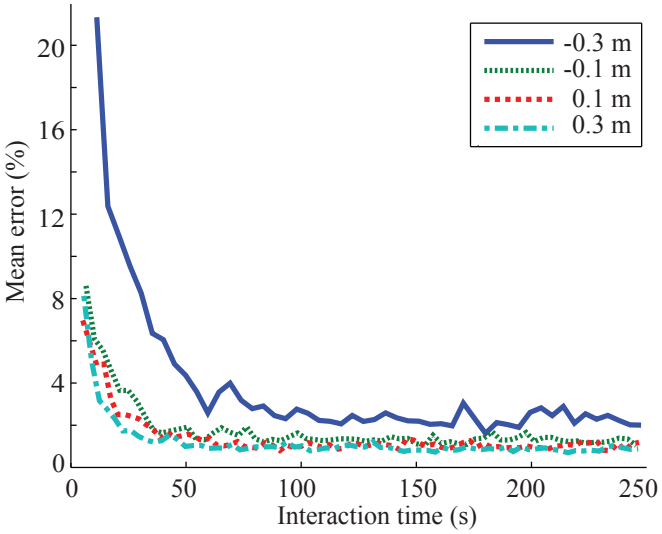


Figure 6.5: SARSA(λ) learning curves for the simulated system, averaged over 30 independent trials. The far left position takes significantly longer to converge than the other targets.

As the proposed velocity-field controller depends on zero-control paths that are sensitive to model inaccuracies, it is expected that the performance is dependent on an *accurate model*. So on this point, it is at a disadvantage when compared to receding horizon controllers, which are known to cope well with model inaccuracies, or reinforcement learning controllers, which require no model at all.

The cost of not using a model in reinforcement learning, is that a motion should be learned which requires a lengthy *learning phase* before operation can start. The velocity field controller requires only the computation of the zero-control paths, whereas the receding horizon controller requires no planning beforehand at all. But, not planning beforehand means the work itself gets harder online. The receding horizon controller requires much more *online computation* than the other methods. The velocity field controller requires a little computation as the correct torque needs to be computed from the velocity field. The reinforcement learning controller requires almost no computation after learning.

The dissipation constraint clearly prohibits some goal states to be reached from some initial states. But how well does the controller fill the space of feasible initial/goal state combinations? This question is particularly important as it directly relates to the robots capabilities to recover from large disturbances. The heuristic behind the velocity field controller is partially aimed at providing such *reachability*, but gives no guarantees. Again, for the other two controllers this reachability is a trade-off with computational costs. Because full reachability for the receding

Table 6.3: Theoretical comparison between the three controllers

Criterion	ZCVF	MCTS	RL
No accurate model needed	−	±	++
No offline computation	±	++	−−
No online computation	+	−−	++
Reachability	−	±	+
Accuracy	−	+	±

horizon controller requires the motion to be planned completely at every time step, the computational cost of reachability is expected to be lower for reinforcement learning than for the receding horizon controller.

Finally, the performance measure that was tested and compared in the experiments: *accuracy of the controller*. In principle, the accuracy of the velocity field controller is low, because it approaches the velocity field in infinite time, whereas the motion takes a finite time. Furthermore, there is an unknown effect caused by projecting the desired torques onto the dissipating torques. For the other two control methods, accuracy can be traded for computational costs. The robot results obtained deviate slightly from this expectation, due to a bottleneck in terms of computational cost: online computation. This bottleneck made it impossible for the receding horizon controller to simulate many variations in the control approach. This is likely the reason the receding horizon controller was less accurate than the reinforcement learning controller, and even less accurate than the velocity field controller.

6.7.2 Future work

From the comparison in table 6.3, obvious directions for improvement of the three controllers can be found. For the velocity field controller, the main challenge is to extend the logic behind following the no-control path into guarantees of convergence. The most promising area here seems to be to look more carefully towards the power flow within the system. The disadvantages associated with the other two controllers are more general, in the sense that they also hold for other types of problems. For the receding horizon controller, it is interesting to look at using a learned model, or to use the intrinsic capacity of the MCTS to deal with stochasticity in order to improve the robustness of the controller. For the reinforment learning based controller, the main improvements will come from designing a more sample-efficient (batch-mode or policy search) algorithm for reinforcement learning.

6.8 | Conclusion

This chapter introduced the problem of dissipatively actuated control of mechanical systems. A type of control intuitively related to steering when skiing. The main problem in these systems is that the actuators can only brake, i.e., take energy out of the system or redistribute it. It must therefore rely on potential energy, for instance from gravity, to power the motion, and on actuation to steer. Such brake-steering can potentially be used to create robots with low energy consumption. This chapter shows that standard model based non-linear control techniques do not suffice for this problem, meaning the dissipatively actuated control problem is in itself an interesting challenge. To tackle that challenge three numerically oriented approaches are proposed which have been tested on a 2-link manipulator, by making it move from one initial position to four different goal positions. In conclusion:

- **ZCVF** A newly proposed velocity field controller that is based on the heuristic of steering towards the path-of-zero-control as quickly as possible when limited control authority is available. This method requires an accurate model, but fairly limited computation both in planning and during motion. In the robot-experiments, an average accuracy of 1.9% of the length of the arm was obtained.
- **MCTS** A receding horizon control that copes with less-accurate model, but requires vast computation during motion. In robot-experiments an accuracy of 3.0% was found.
- **SARSA** A reinforcement learning technique that finds optimal solutions without using a model, but requires interaction with the real system prior to execution. In the robot-experiment the accuracy of this approach was found to be 1.2%.

Acknowledgement

W.J. Wolfslag and M.C. Plooij have received funding from the research programme STW, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO). W. Caarls has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 611909.

7

Learning indirect optimal control for dynamic motion planning with RRT

Wouter Wolfslag, Mukunda Bharatheesha, Thomas Moerland and Martijn Wisse, *Robotics and Automation Letters, Vol 3, 2018.*

Abstract

Sampling-based kinodynamic planners, such as Rapidly-exploring Random Trees (RRTs), pose two fundamental challenges: computing a reliable (pseudo-)metric for the distance between two random nodes, and computing a steering input to connect the nodes. The core of these challenges is a Two Point Boundary Value Problem, known to be NP-hard. Recently, the distance metric has been approximated using supervised learning, reducing computation time drastically. Previous work on such learning RRTs use direct optimal control to generate the data for supervised learning. This chapter proposes to use indirect optimal control instead, because it provides two benefits: it reduces the computational effort to generate the data, and it provides a low dimensional parametrization of the action space. The latter allows us to learn both the distance metric and the steering input. This eliminates the need for a local planner in learning RRTs. Experimental results on a pendulum swing up show 10-fold speed-up in both the offline data generation and the online planning time.

7.1 | Introduction

For motion planning of robotic manipulators, kinodynamic planning and sampling-based planning are getting increasingly popular. Kinodynamic planning, i.e., planning in state space rather than configuration space, improves robustness, speed and energy efficiency of robots [34, 130, 185]. Combining configuration-space planning with post-processing in state-space is often successful [126], but cannot solve all challenging dynamical problems. Sampling based planning has been shown to be the most viable way to handle high dimensional spaces and obstacles [70, 93]. Therefore, this chapter will consider how to apply Rapidly-exploring Random Trees (RRTs) [93], the most popular sampling-based single-query planning algorithm, directly to state space planning for deterministic, fully modeled systems.

RRT builds a tree graph structure with the states of the system as nodes and the trajectories of the system between two states as edges. The algorithm selects a node to expand from based on a *distance* to a randomly sampled node in the state space, i.e., it selects the *nearest* node in the current tree. That node is then expanded by means of a local planner that aims to reach the randomly-sampled node. These steps happen online, so computation time is crucial. Unfortunately, in state-space, both local planning and distance computation are computationally expensive [93].

In literature, the main approach to reduce the computational burden is to approximate the true distance function by a heuristic [56, 76, 125, 154]. Two frequently

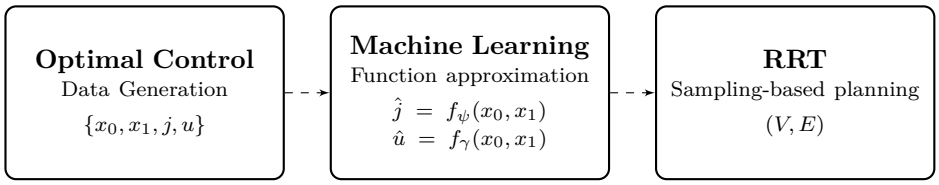


Figure 7.1: Schematic of the Learning-RRT architecture. First, optimal control generates a dataset that specifies the local steering cost j and control input u for a given start state x_0 and end state x_1, Subsequently, machine learning predicts the steering cost and input given a start and goal node. These first two (computationally intensive) phases happen *offline*. Subsequently, the RRT handles *online* planning. The function approximators provide the RRT with quick cost and input prediction without online optimization.

used heuristics are the Euclidean distance and the optimal steering cost for a linear approximation to the system. The convergence of RRT variants using the Euclidean distance heuristic, and random steering inputs, was extensively analyzed in [95]. For promising results for the linearizing heuristic see [58, 151, 178]. However, these heuristics only minimally utilize the dynamical properties of the system, and therefore typically require more nodes to solve a given problem using RRT.

Recent literature proposes a promising different approach, which we call Learning-RRT [4, 16, 122]. Learning-RRT involves an offline machine learning phase that learns the distance and steering function in RRT (Fig. 7.1). An optimal control algorithm provides a database of optimal trajectories, which is the input for a supervised learning algorithm. This algorithm learns to approximate the functions the RRT requires. Note that trajectories found by RRTs are in general not optimal, even if the segments in the database are. Using optimal segments in the database provides a meaningful steering-cost metric and similarly shaped trajectories for connecting similar points in state space, which helps the learning algorithm. Supervised learning provides two benefits: 1) generalization over state-space and 2) fast online predictions. Thereby, the computations for trajectory optimization does not have to be repeated for new situations, and is shifted offline.

This chapter proposes a method that helps to overcome the two remaining challenges of Learning-RRT:

1. **Local planning**: In literature on learning-RRT [16, 122], only the distance function is approximated by machine learning. Supervised learning of the steering function is hard due to the large number of parameters typically required to describe optimal input signals. Therefore, previous Learning-RRTs resort to computationally expensive optimizations for their steering function [16].

2. **Dataset generation**: The dataset for the supervised learning algorithm consists of many optimal trajectories. As optimizing a single trajectory already is a significant computational burden, generating a full dataset is very computationally demanding.

The main contribution of this chapter is the use of indirect optimal control to generate the dataset. This allows us to solve both the problems mentioned above, thereby decreasing the computational burden by up to two orders of magnitude, both in the offline and the online phase of the learning RRT planning. However, the dataset generated by this method contains a bias, which is problematic for the learning algorithm. It turns out we can efficiently remove this bias through a simple dataset cleaning algorithm.

The focus of this chapter is the introduced method and its implementation details. Our experiments provide proof of concept by evaluating our method on a pendulum swing-up, as was done in [16]. While simple, this task allows us to demonstrate and compare the validity of our method. To our knowledge, we are the first to demonstrate learning of the control input in a kinodynamic sampling-based planner.

The structure of this chapter is as follows. Sec. 7.2 describes the Learning-RRT algorithm. This algorithm is applicable to any data generation method, and intended to structure all Learning-RRT components. The subsequent sections combine to introduce RRT-CoLearn, which is a Learning-RRT. The main contribution comes in Sec. 7.3, which tackles the problems of *local planning* and *dataset generation* via indirect optimal control. The class of systems for which this approach is valid can be found in that section as well. Sec. 7.4 discusses how to remove the dataset bias introduced by optimal control. In Sec. 7.5, we discuss how the learned steering-cost metric affects the convergence of the RRT algorithm. Sec. 7.6 presents experimental validation of our algorithm. Finally, Secs.7.7 and 7.8 contain discussion and conclusions.

7.2 | Learning-based RRT

Learning-based RRTs leverage (supervised) learning to speed up the computationally expensive modules of a kinodynamic RRT. The Learning-RRT algorithm is presented in Algorithm 0. Here we present the standard, forward search RRT version of the algorithm as used in the experiments. Enhancements are briefly discussed in Sec. 4.4.

The first step in the algorithm is to create a dataset of optimal trajectories through the state-space of the system ($\mathcal{X}$). This step disregards obstacles, as they are handled later on. The dataset $D = \{b^i\}_{i=1}^N$, has entries $b^i = \{x_0^i, x_1^i, j^i, u^i\}$ that

Algorithm 7 Learning RRT ((V, E), N)

$\hat{D} \leftarrow$ `generate_data`(N) // Section 7.3
$D \leftarrow$ `clean_data`$(\hat{D})$ // Section 7.4
$\hat{J} \leftarrow$ `fit_cost`(D)
$\hat{U} \leftarrow$ `fit_input`(D)
$\hat{V} \leftarrow$ `fit_valid`(D) // Section 7.5
$(X, E) \leftarrow (x_{\text{initial}}, \emptyset)$
solutionfound $\leftarrow$ False
while NOT(solutionfound) **do**
  $x_{\text{target}} \leftarrow$ `sample`()
  if $ANY(\hat{V}(x, x_{\text{target}}))\forall x \in X$ **then**
    $x_{\text{nearest}} \leftarrow \arg\min_{x \in X} \hat{J}(x, x_{\text{target}})$
    $(c, x, u) \leftarrow$ `simulate`$(x_{\text{nearest}}, \hat{U}(x_{\text{nearest}}, x_{\text{target}}))$
    if not `collision`(x_{nearest}, x) **then**
      $X \leftarrow X \cup \{x\}$
      $E \leftarrow E \cup \{(x_{\text{nearest}},\ x,\ c)\}$
return X, E

consists of an initial state $x_0 \in \mathcal{X}$, a final state $x_1 \in \mathcal{X}$, a distance metric/local steering cost $j \in \mathbb{R}^+$, and a set of parameters $u \in \mathcal{U}$, that describe the optimal input leading the system from state x_0 to state x_1. Note that $\mathcal{U}$ can take many forms, depending on the discretization used. For example, it can be the coefficients of a polynomial or the values at the switch times of a piecewise-linear function.

The optimal trajectories are generally found using a numerical algorithm searching for local optima, which poses a challenge for the supervised learning algorithm. If two solutions are nearby in x_0 and x_1, but hail from a different local optimum, a deterministic supervised learning algorithm (for example trained on mean-squared error) will predict the average over the two solutions. This not only makes the cost prediction inaccurate, but most importantly it ruins the steering input prediction: the average of the two steering inputs in the data could lead to a completely different state than the target state. This local-optimum bias requires us to create the dataset D in two stages. The first stage creates a dataset $\hat{D}$ of size N which contains local-optimum-bias, and is indicated in Alg. 0 by the function `generate_data`. The second stage `clean_data` removes the local-optimum-bias to create the desired dataset D.

The second step is to use a supervised learning algorithm on D to approximate the two functions that define the optimal control solution: 1. the function $\hat{J} : (\mathcal{X}, \mathcal{X}) \to \mathbb{R}^+$, which maps from an initial and a final state to a steering cost, 2. the function $\hat{U} : (\mathcal{X}, \mathcal{X}) \to \mathcal{U}$, mapping the initial and final state to the required input parameters.

For both function approximators we implement k-nearest neighbours [50], a standard non-parametric function approximator with robust performance in smaller state-spaces. For a test point x, we identify the k nearest neighbors in our dataset D. The predicted value (e.g. for cost j) for the test point is the average value of these neighbors. We cover extensions to other learning techniques in Sec. 4.4.

The next step, `fit_valid`, addresses an inherent limitation of supervised learning and is treated in Sec. 7.5.

The fourth stage of the Learning-RRT algorithm, the online RRT stage, handles long-term planning and obstacles. First, it samples a point and tests for a valid connection from any node in the tree to the sampled point. If that exists, it expands the node that is nearest to the sampled point according to the function $\hat{J}$. The expansion will use the learned steering function $\hat{U}$, which likely makes a small error. The new node is therefore not exactly at the sampled state. The trajectory to the new state is checked for collisions. If collision free, the new node is added to the tree. The algorithm iterates these steps until it connects to the desired region in state-space. As standard in RRTs, the sampling of new nodes is biased towards the goal by intermittently replacing the sampled state with a desired end-state.

7.3 | Data generation

The dataset for the function approximator is generated from a set of optimal trajectories. The most common approach to find these trajectories are the so-called *direct* optimal control approaches [136, 167]. In these approaches the state equations and cost function are approximated by a discretized system, which is then numerically optimized.

An alternative to direct optimal control is the much older *indirect* approach [135, 142], which first optimizes and then discretizes. For many applications, direct approaches replaced the indirect approach due to better numerical stability at long planning distances. However, it turns out indirect optimal control is ideally suited for the RRT scenario. First, the numerical instability poses no problem for the short segments that are required for RRT. Furthermore, indirect optimal control brings two important benefits. First, it parametrizes the control input in a low dimensional space, which allowing it to be learned. Second, it removes the need for optimization in the sampling process, which speeds up data generation. Both benefits are further explained at the end of this section, after the indirect optimal control method is described. At that point, we will also explain the remaining downside of the indirect optimal control approach: a more biased dataset.

Indirect optimal control

Here the indirect optimal control procedure is used to derive the optimal equations of motion for a pendulum swing-up. The procedure can be applied to other systems, with small differences. For more details and proofs see [117].

Indirect optimal control finds the functions $x(t)$ and $u(t)$ from time $t \in \mathbb{R}$ to state $x \in \mathbb{R}^n$ and input $u \in \mathbb{R}^m$, that minimizes a cost function of the following form:

$$J(x(t), u(t)) = \int_0^{t_\mathrm{f}} C(x(t), u(t))\mathrm{d}t \tag{7.1}$$

$$\begin{aligned} \text{Subject to:} \quad & \dot{x}(t) = f(x(t), u(t)) \quad \forall t \in (0, t_\mathrm{f}), \\ & x(0) = x_\mathrm{initial}, \quad x(t_\mathrm{f}) = x_\mathrm{final} \end{aligned} \tag{7.2}$$

where x_initial and x_final are fixed initial and goal states, and the final time t_f is optimized along with $x(t)$ and $u(t)$. We will often drop the explicit dependency on the time t.

For the pendulum we get $f(x, u) = (\omega, \sin(\theta) + u)$, with $x = (\theta, \omega)$, θ and ω the (angular) position and velocity respectively, and u the torque. The cost integrand $C = 1 + u^2/2$, which takes into account both the time it takes to reach the goal-state as the control effort to do so.

The first step in the indirect optimal control approach is to define the Hamiltonian $\mathcal{H}$, the sum of the integrand C and the inner product of the state derivatives with a vector of Lagrange multipliers, also called the costate. With $(\lambda_\theta, \lambda_\omega)$ as costate, we obtain:

$$\mathcal{H}(x, \lambda, u) = 1 + {}^1\!/_2\, u^2 + \lambda_\theta \omega + \lambda_\omega(\sin(\theta) + u) \tag{7.3}$$

The second step is finding an optimal input u^*, by minimizing the Hamiltonian with respect to the input:

$$u^* = \arg\min_u \mathcal{H} = -\lambda_\omega \tag{7.4}$$

The third step takes partial derivatives of the optimal Hamiltonian, which is created by replacing the input with the optimal input: $\mathcal{H}^*(x, \lambda) = 1 + \lambda_\theta \omega + \lambda_\omega \sin(\theta) - {}^1\!/_2 \lambda_\omega^2$. Note that this equation only depends on the state and costate. The partial derivatives form a system of ordinary differential equations (ODEs) specifying the evolution of the optimal state and costate over time:

$$\dot{\theta} = \frac{\partial \mathcal{H}^*}{\partial \lambda_\theta} = \omega \qquad \dot{\omega} = \frac{\partial \mathcal{H}^*}{\partial \lambda_\omega} = \sin(\theta) - \lambda_\omega \tag{7.5}$$

$$-\dot{\lambda}_\theta = \frac{\partial \mathcal{H}^*}{\partial \theta} = \lambda_\omega \cos(\theta) \qquad -\dot{\lambda}_\omega = \frac{\partial \mathcal{H}^*}{\partial \omega} = \lambda_\theta \tag{7.6}$$

The last step is to use the Eqs. 7.5-7.6 to find the optimal trajectory towards a desired state. For a given costate, the above equations are (numerically) integrated,

which results in a locally optimal state trajectory. This trajectory depends on the choice of initial costate and the time duration of the integration. The initial costate and final time are tuned to find a locally optimal state trajectory that reaches the desired state. This tuning requires a numerical method that minimizes the difference between final state and desired state. Later we show that this tuning can be avoided when generating the database.

Optimal control problems solved for Learning-RRTs typically have a free final time and a cost integrand C that does not explicitly depend on time. On this type of problem, the constraint that sets the final time can be rewritten as a constraint on the initial costate [117]:

$$\mathcal{H}^*(x(0), \lambda(0)) = 0 \tag{7.7}$$

The main reason why *indirect* optimal control is largely replaced by direct methods, such as multiple shooting, is that the resulting differential equations are unstable, and therefore difficult to numerically solve reliably. However, because a learning-RRT database only requires short trajectories, this instability is not as important.

Benefits of using indirect optimal control

There are two major advantages to using indirect optimal control for Learning-RRTs. First, the input directly follows from the costates; in principle, there is a map $U(x_{\text{initial}}, x_{\text{final}}) \rightarrow (\lambda_\theta(0), \lambda_\omega(0), t_\text{f})$, from initial and final state to a set of only three parameters that describe the input function. This set is small, and will only grow linearly with the size of the state space (and is independent of the number of inputs). In contrast, direct optimal control approaches require to parametrize inputs as functions over time, which results in much larger spaces of parameters. The reduction in the number of parameters means the input function can be learned efficiently, thereby solving the problem of *local planning* in RRTs as identified in the introduction.

To see the second major advantage, look at how the whole dataset is created. In current learning RRTs [16, 122], the dataset is generated by sampling from the $(x_{\text{initial}}, x_{\text{final}})$-space, and then, find an optimal trajectory and cost for each sample. Note that every combination of initial state, initial costate and final time produces an optimal trajectory for a certain final state. So, if we sample from the allowed initial states, initial costates and final times, we effectively sample over all initial state - final state combinations. Thus, we can eliminate the need for numerical optimization by sampling the costate, meaning the data are generated much faster.

The data generation procedure is outlined in Alg. 0. The functions *random_State*, *random_Costate*, and *random_Time* sample random states, costates and final times depending on the problem domain and constraint Eq. 7.7. The function

Algorithm 8 generate_data($N, T_{\text{final}}, r_{\text{bound}}$)

Optimal_ODEs ← Eqs. 7.1-7.6
$\hat{D}$ ← empty()
for $n = 1 : N$ **do**
  x_{initial} ← random_State()
  λ_{initial} ← random_Costate() s.t. Eq. 7.7
  T_{final} ← random_Time()
  x_{final}, J ← integrate(Optimal_ODEs,x_{initial} ,λ_{initial},T_{final})
  append($\hat{D}$,{x_{initial},x_{final},J,λ_{initial}})
return $\hat{D}$

integrate numerically integrates the optimal control equations until the final time (T_{final}) or until the distance from the initial state reaches a reachability bound (r_{bound}). To optimize the efficiency of the data generation we add the intermediate integration results to the dataset, causing dependencies between the data-points. We observed that the learning algorithm performed well despite these dependencies.

In this section, we outlined the indirect optimal control approach to *data generation* in learning RRTs. Because the optimal control algorithm incorporates *costates*, we name the overall algorithm RRT-CoLearn. Its two major advantages are: learning optimal steering inputs (online speed) and generating data without optimizing (offline speed).

7.4 | Dataset cleaning

The dataset generated by Algorithm 0 originates from a search for local optima, and can therefore include a bias that interferes with learning performance. The problem is illustrated for with an artificial dataset in Fig. 7.2 (top), where in the middle input region we have the global optimum at the bottom, but there are local optima above it. A standard function approximator (with squared loss) for a given point input (independent variable) predicts the conditional expectation of the dependent variable, shown in green. Note that the predicted function deteriorates in the middle segment, where the average is predicted instead of the optimal cost.

This is problematic for predicting the cost function and especially harmful for predicting the control parameters. Averaging over two locally optimal control inputs by no means guarantees that we end up anywhere close to the target. As an intuitive example, consider the Dubins vehicle, which can be seen as a model of a non-holonomic car. The vehicle can reach a goal behind it by either steering left or right, but will fail to reach the target when taking the average (straight ahead). To counteract this issue, we need a dataset cleaning algorithm, i.e., a procedure that

Algorithm 9 clean_data($\hat{D}, d, k_{\max}$)

$D \leftarrow \hat{D}$
$k \leftarrow 0$
while $k < k_{\max}$ **do**
  $p_{\text{sample}} \leftarrow \text{selectRandom}(D)$
  $p_{\text{neigh}} \leftarrow \text{nearestNeighbor}(p_{\text{sample}}, D)$
  if $\text{distance}(p_{\text{sample}}, p_{\text{neigh}}) < d$ **then**
    $k \leftarrow 0$
    $p_{\text{high}} \leftarrow \arg\min_{p \in \{p_{\text{sample}}, p_{\text{neigh}}\}} \text{Cost}(p)$
    $\text{remove}(D, p_{\text{high}})$
  else
    $k \leftarrow k + 1$
return D

somehow eliminates conflicting (non-optimal) datapoints. In literature, there are resampling methods for dataset imbalance, most noteworthy class label imbalance in classification tasks [52]. However, our dataset is not imbalanced, but rather contains a systematic bias. It turns out we can leverage that structure to come up with a simple resampling/cleaning algorithm.

For each point in input space, we are interested in retaining the lower bound of the cost of the generated datapoints. First note that we prefer to remove points in high-density regions, as in low-density regions there is little to throw away, and we may only hope that our data are accurate. We implicitly remove from high density regions by first uniformly sampling a point from our dataset. We then search for its nearest neighbor in the dataset based on Euclidean distance. If this neighbor is within a distance d from our sampled point, we remove the node of the two with the highest cost, frequently removing a biased data-point while retaining a good one. Otherwise, we retain both points, which will happen in low density regions. This process is repeated until no points are removed for k_m consecutive steps, after which we return the cleaned dataset. The procedure is outlined in Alg. 0.

The main parameter in this algorithm, d, can be interpreted as a neighborhood size. Fig. 7.2 shows it has an optimal setting that depends on the dataset and which has to be found empirically. To evaluate it d, we fix it, run the cleaning, fit the function predictors, and then sample new datapoints to assess the error in cost and co-state parameters on the predictions. This evaluation is not ideal, but the ideal metric (RRT performance) is too expensive to compute.

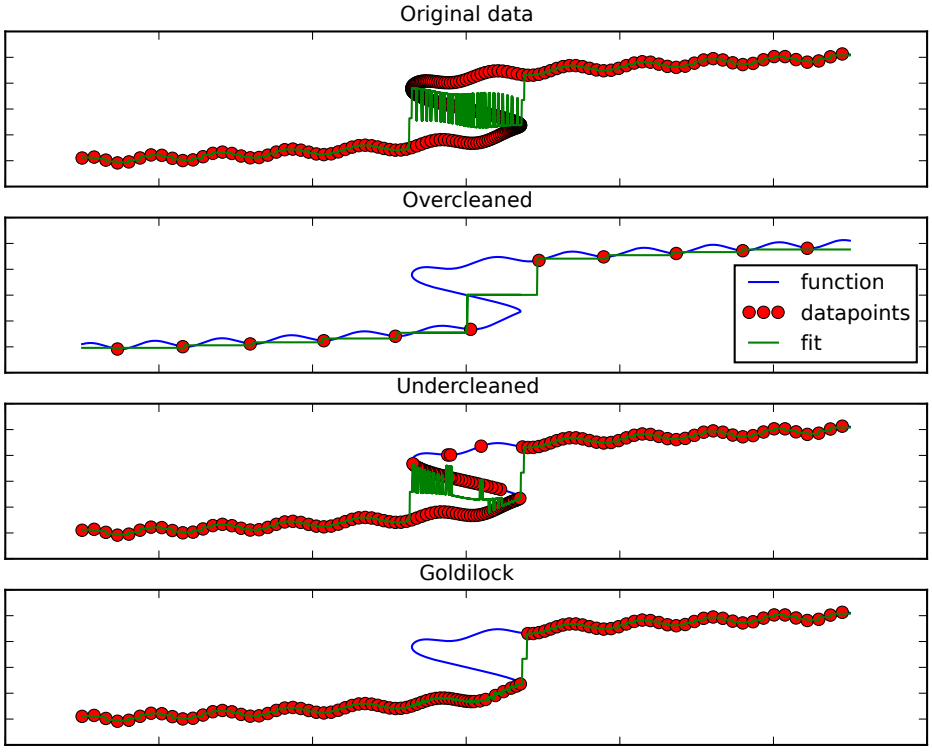


Figure 7.2: The data-bias problem and the effect of the d parameter in the data cleaning algorithm. The top figure shows an imaginary dataset, which has a problem with bias in the middle of its domain. The fitted function is a poor approximation of the least cost part of the datapoints. In the second figure d was chosen too large: the bias is gone, but there is not enough resolution left to accurately fit the function. In the third figure, d is too small: not all bias is removed. The bottom figure shows a proper choice for d: the bias is removed, and enough resolution remains.

7.5 | Notes on completeness

The machine learning approximations in the proposed algorithm intuitively might interfere with its completeness properties. In this chapter we do not prove probabilistic completeness of learning RRTs or RRT-CoLearn. However, we made a number of small additions that improve the algorithm and make it more amenable to a future proof of convergence. This section discusses these additions, and the reasoning behind them. A future proof of completeness will likely follow the lines of earlier work such as [75, 93, 95].

For the first addition, note that our inputs are generated based on a prediction of the costate from the function approximator. If the input is constructed directly based on this prediction, the rest of the inputs will have no chance of being selected.

If the prediction is not perfectly accurate, this could cause the algorithm to get stuck. Therefore, the input resulting from the prediction is generated based on a probability distribution that associates a high probability of choosing a value close to the prediction, while giving some probability to all possible inputs. In our experiments, the control parameters are samples from truncated normals, with the bounds for each parameter specified by its sampled domain. The means are the value predicted by the learned model. The standard deviation σ of the (non-truncated)-normal is a parameter of the algorithm.

To make our algorithm more similar to the algorithm discussed in [93], which might make it easier to adjust the proof found there to our algorithm, we discretized the action space. This was done by rounding the input to two decimals. As this is a relatively fine discretization, we expect it had little effect on the experiments.

Experiments showed that inaccurate prediction makes selecting the right input difficult when the problem requires the RRT to very precisely reach a small region in state space, as happens when near the goal region. Therefore, when the target in an RRT step involves the goal region, we increase the standard deviation of the input distribution. This empirically improved performance.

Errors made by the distance function approximation could also cause the algorithm to get stuck. If the distance towards a certain node is always underestimated, other nodes might no longer be expanded. Alternatively, if the distance to a node is overestimated, that node might no longer be expanded.

The more dominant method to prevent approximation errors in the distance function to hinder the algorithm looks at a specific type of error. Learning algorithms are intended to generalize using interpolation, so may make large errors when extrapolating. This is especially true for Learning RRTs, for which this problem has not been identified in literature yet. In RRTs we *uniformly* sample state-space, while we have confined our dataset to only contain short motion segments. Therefore, sampled combinations (x_0, x_1) are often outside the dataset, where function approximation may make large errors. Notably, the approximated distance metric might greatly underestimate the steering cost from a certain node, causing that node to be incorrectly chosen for expansion.

This issue can be solved by a binary classifier that decides whether a query would yield a valid prediction, i.e., whether the dataset D covers the queried point. The implement classifier $\hat{V} : (\mathcal{X}, \mathcal{X}) \to \mathrm{v}$, with $v \in [\texttt{true}, \texttt{false}]$, simply computes the summed distance to the nearest neighbors of the queried point to the points in the dataset, and rejects the query point if this sum becomes too large. An alternative approach, explored in [154], relies on reachability: the notion that the dataset contains only short segments, meaning the final states should be reachable within a short period of time.

The second method to limit the effect of approximation errors is by directly excluding overly small or large values for the distance. In our experiments, it was implemented by clipping the distance value to $10^{\pm 5}$ times the Euclidean distance, and had little effect on the computation.

7.6 | Experiments and results

To test our approach, we perform experiments on the pendulum described in Sec. 7.3. The task is to move the pendulum from its stable equilibrium $(\theta, \omega) = (-\pi, 0)$ to its unstable equilibrium $(0, 0)$.

This experiment does not require obstacle handling; we do not explicitly focus on collision checking or obstacles in our work, as our focus is on the RRT itself and we assume the collision checking is handled by an external algorithm.

Data were generated and cleaned for separate epochs, with 300 runs of the RRT algorithm per epoch. The data for each epoch consist of 40000 simulations, which ended when the local cost exceeds 2 or when the norm of the state difference with the initial state (r_{bound}) exceeds1.5. Integration was done by the 4th order Runge-Kutta algorithm with a time step of 0.01 s. The initial position was uniformly sampled from $(-3\pi/2, \pi/2)$rad, the initial velocity from $(-\pi, \pi)\text{rad}\,\text{s}^{-1}$, and the initial costate sampled as described below. The data cleaning resolution d equals 0.05. The data cleaning stopping parameter $k_{\max}$ is set to 5000. The nearest neighbor fitting algorithm during the RRT takes $m = 3$ nearest neighbors. Finally, the standard deviation of the sampling distribution $\sigma = \pi/4$ normally, and $\pi/2$ when the query involves the goal state. The experiments ran on a MacBook with Intel(R) Core(TM) i5-3210M 2.50GHz CPU and 8GB of RAM, running Ubuntu Linux 14.04 and code in Python and Julia.

To avoid projecting on the costate constraint (Eq. 7.7), which is computationally expensive for larger systems, the constraint is solved explicitly by uniformly sampling the parameter $\phi \in (-\pi/2, 3\pi/2)$ that sets the initial costate as follows: $\lambda_\theta = \tan(\phi)$ and $\lambda_\omega = -\sin(\theta) + \text{sign}(\cos(\phi))\sqrt{\sin(\theta)^2 + 2 + \tan(\phi)\omega}$. If λ_ω has an imaginary part, the simulation is disregarded.

Figure 7.3 shows a typical run of the algorithm. Three important points can be seen from this figure. First, the algorithm neatly expands through state-space. While it favors expanding along the circular paths that correspond to low input trajectories, it expands nodes in all feasible directions. This indicates that the distance metric works properly, and contrasts with trees that are grown without appropriate distance metric, which tend to have many nodes clustered. Secondly, the figure highlights the approximation errors that still exists; the edges that lead to the border of the figure all had target nodes inside the figure. As these nodes are outside the figure, the target was not reached exactly. Finally, the algorithm

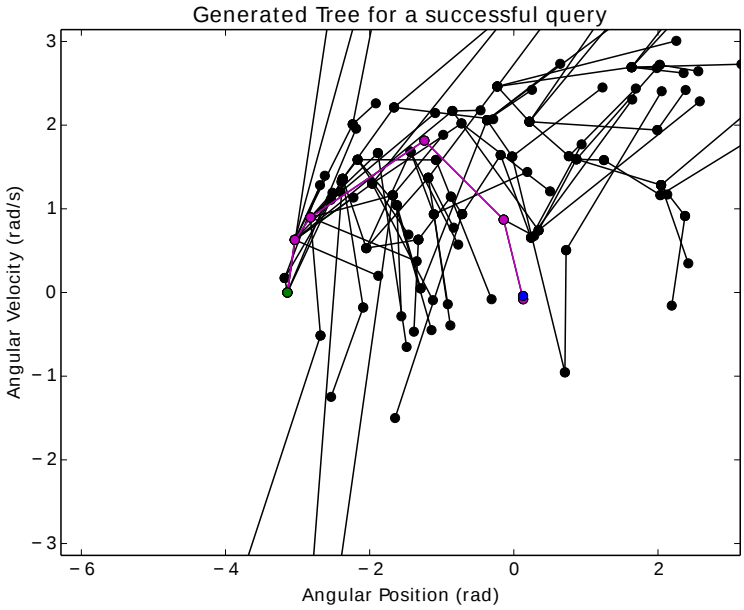


Figure 7.3: The state-space coordinates of the tree-nodes of a successful run of the algorithm. The edges are shown as straight lines, with those connecting the initial point to the goal being highlighted.

has tried to reach the goal-state a number of times from the same node. This shows the effect of the increased randomness; because the attempts are spread out sufficiently, the goal is eventually reached.

Figure 7.4 shows the computation times for 10 epochs. The planning time has the same variation in each epoch, indicating that data generation and cleaning are performed robustly. The median time to reach the target over all samples was 2.43 s, over 10 times faster than [16] on the same hardware. Data generation and cleaning took ∼25 min per epoch, an order of magnitude faster than [16][1].

The performance of the learned steering function is assessed by the mean squared error between the target state and the final state attained by using the predicted costate. The median of this error over all epochs is 0.11.

The quality of the combined machine learning is assessed using the number of nodes needed to reach the target. For the results in Fig. 7.4, the median over all runs is 84 nodes. About 30% smaller (better) than in [16], this result is best interpreted as a roughly equal performance, as the problem here does not require the pendulum

[1]The cited paper does not report the offline computation time. However, due to overlapping authors we know offline computation took nearly a week.

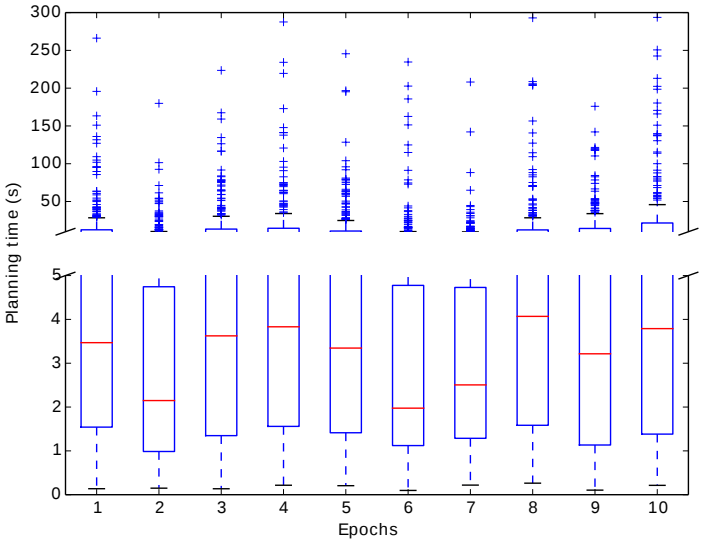


Figure 7.4: Boxplot of planning times for ten epochs.

to swing back and forth to reach its goal. Equal performance indicates the learned steering function approximates the online optimization well.

Figure 7.5 shows the effect of the reachability bound on the number of nodes needed to reach the goal position. This investigates how the length of the simulation in the dataset influences the performances of the algorithm. When the simulations are too short, the online algorithm is forced to use many segments, hindering convergence. When simulations are too long, the coverage of all possible trajectories becomes sparse, causing poorer learning performance.

Figure 7.6 shows the number of nodes needed to reach the goal position with various parts of the algorithm removed or replaced. This allows us to identify the important aspects of the algorithm, and to compare the algorithm to the state-of-the-art. The last four algorithms have the validity check turned off and perform worst. Note that [16] does not use such a check, but instead implements an online trajectory optimization to avoid the poor performance without validity check shown here. The figure also shows that the difference between learned and Euclidean distance is small when using a validity check, confirming the intuition from [154]. The algorithm in that paper is very close to the Δ___-algorithm in Fig 7.6. Finally, we see that learning the actions makes the algorithm perform better than using random costate selection.

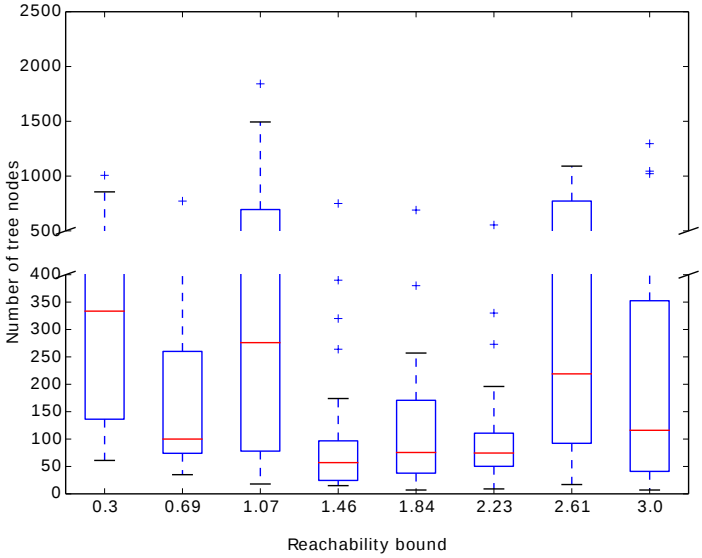


Figure 7.5: The number of nodes required to find a solution using different settings of the reachability bound.

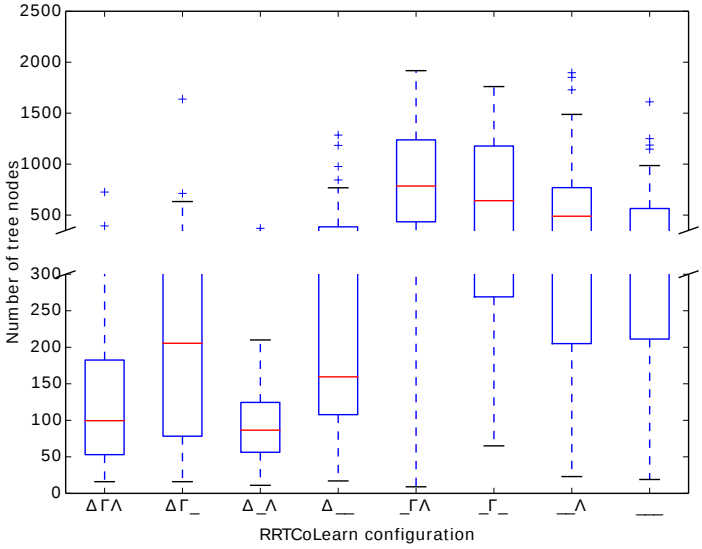


Figure 7.6: The number of tree nodes required to find a solution for different configuration settings for RRT-CoLearn algorithm. The horizontal axis is labeled with three symbols, which can be turned on or off. In the first position, the Δ signifies the validity check being turned on. Similarly, the Γ switches between learned and Euclidean distance, and the Λ between learned and random costate (action) selection.

7.7 | Discussion

The algorithm introduced in this chapter allows learning of not only the distance metric, but also the steering input. As proof of concept, we tested our algorithm on a basic pendulum swing up problem. It reduces the time spend both in the offline learning and in the online computation by a factor of more than 10, a large step towards sampling based kinodynamic planning in a practical setting.

The main direction for future research is to support higher degree of freedom systems. Such systems will require more sample efficient data generation and learning. To improve the efficiency of data generation, new simulations might be selected based on already obtained data, and the partially learned cost and steering functions, similar to what has been done for reinforcement learning [94]. (Deep) generative models could provide more efficient learning. These models allow sampling from complex, high-dimensional probability distributions, meaning we could retain all data points without averaging over solutions [57, 112].

A secondary direction for future research is to support input bounds. We have taken a preliminary step in this direction by implementing RRT-CoLearn on a time optimal pendulum swing up with the torque bound set to 0.5, and all other constants as in Eq. 7.3. The resulting optimal equations of motion are invariant to the norm of the costate vector. Therefore the constraint of Eq. 7.7 is satisfied by setting the norm of the costate to 1. For 50 runs of RRT-CoLearn, the solution was always found within 2000 nodes, using a median planning time of 12.32 s, and median number of 195 nodes. These results compare favorably with those reported by the state-of-the-art in kinodynamic planning [16, 126, 154].

Unfortunately, the input constraints can cause an exact overlap between trajectories starting from different costates, at least for a finite time. Such overlapping trajectories are difficult for the learning and cleaning algorithms we used. Therefore, a larger dataset was required, severely slowing down the RRT. Extending the machine learning aspects of RRT-CoLearn, such that they can cope with overlapping trajectories is an important theoretical and practical issue.

The third direction for future research is the combination of the RRT-CoLearn algorithm with other (non-learning) enhancements of the basic RRT algorithm, such as [82, 90]. RRT-Connect [90] would be the most prominent addition. It grows two trees: one forwards from the initial state, and one backwards from a goal state. A new model based on backwards integrated data must be learned for creating the backwards-searching tree. As the system of differential equations (Eqs. 7.5-7.6) is unstable in both forward and backward direction[117], the learning challenges remain similar.

Finally, note that RRTs, and therefore learning RRTs, are suited to finding novel motions for deterministic systems for which an accurate model is available. When

	Optimal Control	Machine Learning	RRT
+	Distance computation Optimal local planner	Generalization Fast online prediction	High dimensional Obstacle avoidance
-	Costly computation[7.3] Local optima→bias[7.4]	Needs large dataset[7.3] Needs unbiased data[7.4] Bad extrapolation[7.5]	Needs distance metric Needs local planner[7.3]

Table 7.1: Benefits and challenges of components of Learning-RRTs

planning for non-deterministic or uncertain systems, funnel based approaches are more appropriate than trajectories [168]. If the system has to perform similar motions repeatedly, multi query methods such as probabilistic roadmaps [55] or motion libraries can be more effective [15]. For the last problem, learning has already been used to find similarity between obstacle locations between planning instances [166]. It is interesting to investigate if using indirect optimal control similarly to this chapter could be beneficial for those settings.

7.8 | Conclusion

This chapter first described a general Learning RRT algorithm. Table 7.1 shows the benefits and challenges of its components: Optimal Control, Machine Learning and RRT. The superscripts in the table refer to the sections in which the challenges are addressed.

RRT-CoLearn, an instance of a learning-RRT, was proposed. RRT-CoLearn generates data faster and allows learning of the steering function, both by using indirect optimal control. It also uses a newly proposed data-cleaning algorithm for more accurate function approximation.

RRT-CoLearn was successfully used on a pendulum swing up both with and without input constraints. The main results are on the system without input constraints and show that RRT-CoLearn is 10 times faster than a state-of-the-art learning RRT [16].

8

Two notes on RRT-CoLearn: a proof of probabilistic completeness and a general sampling procedure

Wouter Wolfslag, Mukunda Bharatheesha,
A paper that is based partially on the contents of this chapter is currently under review at
IEEE International Conference on Robotics and Automation.

Abstract

The RRT-coLearn algorithm presented in the previous chapter is not completely developed yet. This chapters answers two questions, bringing the algorithm closer to maturity. The first question is: will the algorithm always find a solution, at least theoretically? This question is answered by proving that the algorithm is probabilistically complete. The second question is: can the data generation part of the algorithm be generalized beyond the pendulum swing up? We show that the data generation can be extended to a large class of robots for three important control costs, two of which include input constraints.

8.1 | Introduction

To approach human quality of movement, robots need to plan their motion through state-space, rather than configuration space only. For a variety of tasks, planning in state-space either greatly improves the resulting motion, or is necessary to make successful completion possible at all[34, 130, 185].

Unfortunately, planning in state-space, also known as kinodynamic planning, is hard. A promising technique is to combine the advantages of optimal control with those of machine learning and those of sampling based planning. A number of papers have proposed similar schemes [4, 16, 122], all based on the Rapidly Exploring Random Trees (RRT) approach to sampling based planning. Such Learning RRTs have been shown to greatly speed up RRTs for kinodynamic planning. In [184], a generic description of learning based RRTs was given, along with an important addition: the use of indirect optimal control as part of the algorithm. The results of this study are reproduced in Chapter 7.

However, while a general description of learning based RRTs was given, the only proof of its performance is based on empirical results on simple systems. As such, it remains uncertain what the performance of the proposed RRT-CoLearn algorithm would be in a more general sense.

This chapter contains two notes on such more general performance. First in Section 8.2, a proof of the probabilistic completeness of the algorithm is given. This establishes that the algorithm is sensible, meaning that when a solution exist, the algorithm will find it when given enough time. Second, in Section 8.3, it is shown how to use indirect optimal control to sample optimal trajectories for a large class of important robotics problems.

8.2 | Probabilistic completeness

The machine learning approximations used in learning RRTs might intuitively interfere with their completeness properties. Therefore, we establish probabilistic completeness of learning RRTs, by modifying the proof for the original RRT [93] in a way that is partially inspired by the proof in [26]. We have not tried to make the proof outlined here as general or rigorous as possible. The framework from [95] might aid in that effort, as might the ideas from [75].

The proof given below is not for RRT-CoLearn, but rather for a more generic algorithm which has properties that RRT-CoLearn can achieve. Section 8.2.3 discusses how these properties are achieved for the RRT-CoLearn algorithm presented in Chapter 7.

8.2.1 Definitions

First, we define the planning problem. The goal of planning is to reach the goal set $\mathcal{X}_{\text{goal}}$, a subset of the statespace $\mathcal{X}$ which is a bounded subset of $\mathbb{R}^n$. Some part of statespace is inaccessible due to obstacles. The part of statespace that is not part of obstacles is called $\mathcal{X}_{\text{free}}$. Finally, to steer the robot from its starting state $x_0 \in \mathcal{X}_{\text{free}}$ to the goal set, it can take actions from a finite set $\mathcal{A}$.

The algorithm proposed to solve this problem is defined as follows. First, initialize a tree of states, with the initial state x_0 as the first node. Then, grow the trees using the following steps.

1. Sample x_{r}, a random state from the uniform distribution over $\mathcal{X}$
2. Determine x_{nn} the nearest node to x_{r} in the tree according to some distance function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.
3. Sample an action a_s, according to some probability function $q(a|x_i, x_r)$
4. Simulate the motion m starting from x_{nn} using a_{s}, to determine x_{new}, the final state of the motion.
5. Verify whether all of the states in m are in $\mathcal{X}_{\text{free}}$. If so, add x_{new}, and an edge from x_{nn} to x_{new}. In the following, the edge is identified implicitly both by the action a_{s}, and the trajectory that leads from x_{nn} to x_{new}.
6. Furthermore check if the added x_{new} is in $\mathcal{X}_{\text{goal}}$. If so, stop the algorithm, otherwise repeat from step 1.

Finally, we set the following conditions on the distance d, the probability density over actions q, and the dynamics of the system:

1. The distance function must always be larger than some positive constant c_{lb} times the Euclidean distance.

$$d(x, y) \geq c_{\text{lb}}||x - y||_2 \quad c_{\text{lb}} > 0 \quad \forall x, y \tag{8.1}$$

   Intuitively this prevents the distance towards a node in the tree to be too low, causing the algorithm to always choose that node for expansion.

2. The set $\mathcal{G}(x)$, has a nonzero volume for all $x \in \mathcal{X}$. Here $\mathcal{G}(x)$ is the largest connected set that includes x and has the distance to each point y in the set, i.e., $d(x, y)$, smaller than some constant positive c_{ub} times the Euclidean distance.

$$\text{Vol}(\mathcal{G}(x)) > c_{\text{v}} \; \forall x \tag{8.2}$$

$$\text{where} \quad \mathcal{G}(x) = \text{argmax}\text{Vol}(\mathcal{S}) \,\text{s.t.} \qquad \mathcal{S} \subseteq \{y | d(x, y) \leq c_{\text{ub}}||x - y||_2\}, \quad x \in \mathcal{S}, \quad \mathcal{S} \text{ is connected}$$

   Intuitively, this is an upper-bound condition on the distance, adjusted to cater for the discontinuity of $d(x, y)$ at $x = y$ that occurs in many steering costs. The upper bound prevents the distance to a node from becoming to high, causing the algorithm to never expand that node.

3. The probability function $q(a|x_1, x_2) \geq q_{\text{a}}$ for some positive constant q_{a} for all $a \in \mathcal{A}$, and x_1 and x_2 in $\mathcal{X}$.

4. The last condition specifies that the same action performed at nearby states lead to similar trajectories. The condition resembles a Lipschitz continuity: if the state of a trajectory starting at x, taking action a (which includes a final time t_f) is denoted by $\gamma(x, a, t)$ for all $t \in (0, t_f)$ then: $||\gamma(x, a, t) - \gamma(x + \Delta x, a, t)|| \leq K||\Delta x||$ for all $x, \Delta x, a$ and t within the bounds set by a. Here K is some constant.

8.2.2 Theorem and proof

For this problem, and these conditions, we prove the following theorem:

Theorem 1 (*Probabilistic Completeness*)**.** *Assume there exists a solution consisting of a finite number of nodes: $x_0, x_1, \ldots, x_n$ and accompanying actions $a_0, \ldots, a_{n-1}$, each with a finite duration, such that any trajectories that abides by the differential and input constraints, and that stays within a distance q_δ from that trajectory is also a feasible solution (we say that the solution has a δ-clearance of q_δ). Then, the probability P of finding a solution approaches one as the number of iterations go to infinity.*

Proof. Consider that x_i is the most advanced waypoint of the solution that is already in the tree. The chance that one iteration of the loop of the algorithm reaches the next waypoint in the solution is factored as:

$$P(\text{reach } x_{i+1}) = P(u_i|\text{expand } x_i)P(\text{expand } x_i) \tag{8.3}$$

Now both factors are strictly positive, i.e.:

- $P(u_i|\text{expand } x_i) \geq q_\text{a}$, which follows from condition 3
- $P(\text{expand } x_i) \geq q_\text{x}$, which follows from Lemma 1 below.

Therefore we get: $P(\text{reach } x_{i+1}) \geq q_\text{a} q_\text{x}$. This means the chance of getting to the next node of the solution will approach one as the number of iterations increase, so at some point the algorithm will get to the next node, and the next one, and so on. Therefore the algorithm will eventually find the solution. □

Lemma 1. *$P(\text{expand } x_i) \geq q_x$, i.e., the chance of selecting x_i, the most advanced waypoint of the solution that is currently in the tree waypoint is always larger than a positive constant q_x.*

Proof. Per step 1 and 2 in the algorithm, the chance of expanding from x_i is the volume of the state space for which that node is the nearest node divided by the total volume of the free state-space:

$$P(x_i) = \frac{\text{Vol}(\{x \in \mathcal{X} | d(x_i, x) < d(x_e, x), \forall x_e \in \mathcal{V} \backslash x_i\})}{\text{Vol}(\mathcal{X})} \tag{8.4}$$

Since the volume of $\mathcal{X}$ is fixed and finite we only need to make sure the numerator is non-zero.

Take the largest ball $\mathcal{B}_\rho(x_i)$ centered around point x_i, such that $c_\text{ub}||x_i - y|| \leq c_\text{lb}||x - y||$ for all y in the ball, and all nodes x in the tree. Based on simple Euclidean geometry, $\rho > 0$. Also, by construction, the intersection $\mathcal{B}_\rho(x_i) \cap \mathcal{G}(x_i)$, with $\mathcal{G}$ defined in condition 2, has positive volume. Per conditions 1 and 2, all points in that intersection are closer (by measure d) to node x_i than to any other node in the tree. Therefore the enumerator from Eq. 8.4 is nonzero, and hence $P(\text{expand } x_i) > 0$.

A nonzero chance for expansion for every iteration does unfortunately not imply that expansion occurs at some point. If the chance for expansion at each step approaches zero fast enough, the chance of never expanding can converge to a finite value. The following paragraphs show that the chance of expansion can be bounded from below by a positive constant q_x.

Note that to make the volume of the space that is closest (by measure d) to x_i, normalized by the volume of $\mathcal{X}$, smaller than some value q_{x}, the nearest node in the tree to x_i should be closer by Euclidean distance than some other value q_{n}. If we pick q_{x} small enough, this means (per condition 4, and the finite number and duration of the edges) that there exist another solution from that nearest node, by following the same actions as in the original solution. Furthermore, the volume of statespace for which either node is the nearest neighbor is always larger than q_{x}. Therefore, the chance of extending either one of the solutions is at least q_{x}.

There are two further considerations to complete the proof. First, the same logic as above implies that if there are multiple nodes within a distance q_{n} of the original x_i node, they all provide their own alternative solution, and their combined nearest-neighbor volume is still always larger than q_{x}. Second, if one of the alternative solutions is expanded, the δ-clearance of that solution might be smaller than that of the original solution. However, due to δ-clearance being defined by an open set around the solution, the alternative solution will still have some positive clearance. Due to the solution consisting of a finite number of steps, a possible limit of 0 clearance is never reached. Therefore, with a little abuse of notation, we can say $P(\text{expand } x_i) \geq q_{\mathrm{x}}$. □

8.2.3 Probabilistic completeness in practice

The proof of probabilistic completeness is given for a slightly different problem and for a subtly different algorithm than those discussed in Chapter 7. This section will discuss those differences and make some further observations about the practical aspects of the algorithm with regards to the proof.

The first point of attention is that the previous chapter started with a continuous action space, whereas the theorem aboves involves a finite action space. This is resolved easily by discretizing the action space, as was already done in the previous chapter. By sampling that discretized space using truncated normal distributions, condition 3 was also satisfied. However, discretizing the action space comes at a price. It is possible that there exists a solution to the planning problem for the continuous action space, which does not exists for the discretization. Therefore discretizing the state-space might cause a failure to find a solution. We believe that some form of probabilistic completeness holds for a continuous action space, but we leave finding it for future work.

The second point of attention is the distance metric: does the approximated optimal steering cost meet conditions 1 and 2? The answer depends on the the cost function used for the optimal control, the system, and the learned approximation. First the cost function: if the cost function is such that there exists actions that do not incur a cost, condition 1 is not met. This happens for instance if the cost is solely the square of the input, which is a frequently studied case.

Secondly, the optimal steering cost is known to be a Sub-Finslerian metric [99]. This implies that the distance from a given state is well behaved within some cone of directions. Given sufficient smoothness of the optimal control cost, this almost implies condition 2. Almost, because the cone does not have volume for all systems. Specifically, systems systems need to be small time locally accessible for the cone to have volume. As such, systems that do not meet that demand, such as underactuated systems, also do not meet condition 2.

Note that the validity check used in Chapter 7 effectively identifies states that are outside the cone in which the steering cost is bounded by a scaled Euclidean distance. States outside that cone cannot be reached with a short trajectory, which means that they are not represented in the database and therefore marked invalid. The validity check can therefore be interpreted as a way to improve the qualitative match between the actual and approximated steering cost.

Finally, approximation errors due to learning could cause conditions 1 and 2 to be violated. Unfortunately, this is difficult to assess. However, the clipping of the distance function, as used in Chapter 7 forces the approximated distance function to conform to these conditions.

8.3 | Generalizing initial costate

Chapter 7 showed how to implement RRT-CoLearn for an energy optimal pendulum swing-up. A main step in that algorithm, is to derive the optimal equations of motion via the maximum principle of Pontryagin [117]. For the type of problem under study, these optimal equations involve a constraint on the initial value of the costate. An ad-hoc approach to solving that constraint for the pendulum swing up was shown. This solution was complex, even for this simple system. As a result, the scalability of this method is not obvious.

In this section, the optimal equations of motion will be derived for three of the most important problems in robotics: energy optimal control, time optimal control with input constraints and energy optimal control with input constraints, all for a generic class of systems: fully actuated open chain robots. Next to deriving the equations of motion, we will also provide a sampling scheme for the initial costate, that handles the constraint on the Hamiltonian.

For the derivation of the optimal equations of motion, we use the

First, the equations of motion. For the class of fully actuated open chain robots, they can be represented by the following differential equation:

$$M(q)\ddot{q} = C(q, \dot{q}) + G(q) + \tau \tag{8.5}$$

Where q are the generalized coordinates, M is a positive definite mass matrix, C represent the convective acceleration terms, G the forces due to potential energy and finally τ are the motor torques. Note that friction losses are neglected.

8.3.1 Energy optimal control

The most frequently used cost term in control involves a squared input term. For robots this corresponds to energy losses due to the winding resistance in the motor. This so called copper-loss is thought to be the main source of energy loss in most robots. For our energy optimal control term, we augment this cost term with another term that penalizes the time it takes to reach the goal:

$$J = \int c_t + \frac{1}{2}\tau^\top \tau \mathrm{d}t \tag{8.6}$$

where c_t is a positive parameter that tunes the relative importance between energy and time costs.

As a result, we get the Hamiltonian:

$$\mathcal{H} = c_t + \frac{1}{2}\tau^\top \tau + \lambda^\top \ \dot{q} + \mu^\top M^{-1}(q)(C(q,\dot{q}) + G(q) + \tau) \tag{8.7}$$

Setting the partial derivatives with respect to τ to zero, gives us the optimal motor torque:

$$\tau^* = -M^{-\top}\mu \tag{8.8}$$

Which in turn leads to the optimal Hamiltonian:

$$\mathcal{H}^* = c_t - \frac{1}{2}\mu^T M^{-1}(q) M{-\top}(q)\mu + \lambda^\top \ \dot{q} + \mu^\top M^{-1}(q)(C(q,\dot{q}) + G(q)) \tag{8.9}$$

The optimal control equations of motion are found by taking the appropriate derivatives, as explained in Chapter 7.

To generate the data for RRT-CoLearn, these optimal control equations of motion are numerically integrated, starting from randomly sampled initial states and costates. This random sampling is made more complex, because for this type of problem, the value of the optimal Hamiltonian is constrained to be 0 at $t = 0$.

For sampling initial states and costates while taking into account that constraint, the following approach is proposed. First, sample the initial states uniformly over their domain (q_0 and $\dot{q}_0$). Second, sample a costate uniformly over the unit sphere, resulting in $\hat{\lambda}_0$ and $\hat{\mu}_0$. The costate used in simulation will be a constant (α) times that unit costate vector. By setting the optimal Hamiltonian (Eq. 8.9) to zero, we get:

$$c_t - \frac{1}{2}\alpha^2 m\hat{u}_0^T M^{-1}(q_0)M-\top(q_0)\hat{\mu}_0 + \alpha\hat{\lambda}_0^\top\ \dot{q}_0 + \alpha\hat{\mu}_0^\top M^{-1}(q_0)(C(q_0,\dot{q}_0)+G(q_0)) = 0 \tag{8.10}$$

This is a quadratic equation, which can be readily solved for α:

$$\alpha = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{8.11}$$

With

$$a = -\frac{1}{2}\hat{\mu}_0^T M^{-1}(q_0)M-\top(q_0)\hat{\mu}_0 \tag{8.12}$$

$$b = \hat{\lambda}_0^\top\ \dot{q}_0 + \hat{\mu}_0^\top M^{-1}(q_0)(C(q_0,\dot{q}_0) + G(q_0)) \tag{8.13}$$

$$c = c_t \tag{8.14}$$

Because c_t is positive, so is c; and because $M(q)$ is positive definite, a must also be positive definite. As a result, there are always two solutions. Due to sampling directions from the unit sphere, we can also sample the negative of every sampled $(\hat{\lambda}_0, \hat{\mu}_0)$, which simply swaps the two solutions as found by Eq. 8.11. As a result, we sample all solutions by sampling from that unit sphere and picking the solution from Eq. 8.11 were the $\pm$ is set to $+$.

8.3.2 Time optimal control with input bounds

If the goal is to move as fast as possible, the energy term in 8.6 drops, and we are left with the following Hamiltonian:

$$\mathcal{H} = 1 + \lambda^\top\ \dot{q} + \mu^\top M^{-1}(q)(C(q,\dot{q}) + G(q) + \tau) \tag{8.15}$$

The problem of moving as fast as possible is only sensible for robots if there are constraints on their inputs. As such, the optimal input is found by minimizing the Hamiltonian subject to the input constraints. In our case, the motor torques are bounded by some constant: $|\tau_i| \leq \bar{\tau}_i \quad \forall i$. This results in the optimal torque: $\tau^* = -\sigma(M^{-\top}\mu) \odot \bar{\tau}$, where $\odot$ signifies the elementwise product, and $\bar{\tau}$ is the vector of bounds on the motor torques.

As a result the optimal Hamiltonian becomes:

$$\mathcal{H}^* = 1 + \lambda^\top\ \dot{q} + \mu^\top M^{-1}(q)(C(q,\dot{q}) + G(q)) - |M^{-\top}\mu|_E \odot \bar{\tau} \tag{8.16}$$

Which gives rise to the optimal equations of motion:

$$\begin{aligned}
\ddot{q} &= \frac{\partial \mathcal{H}}{\partial \mu} = M^{-1}(q)(C(q,\dot{q}) + G(q) - \sigma(M^{-\top}\mu) \odot \bar{\tau}) \\
\dot{\lambda} &= -\frac{\partial \mathcal{H}}{\partial q} = -\mu^{\top} \frac{\partial M^{-1}(q)(C(q,\dot{q}) + G(q))}{\partial q} + \sigma(M^{-\top}(q)\mu) \odot \frac{\partial M^{-\top}(q)\mu}{\partial q} \odot \bar{\tau} \\
\dot{\mu} &= -\frac{\partial \mathcal{H}}{\partial \dot{q}} = -\lambda^{\top} + \mu^{\top} M^{-1}(q) \frac{\partial C(q,\dot{q})}{\partial \dot{q}}
\end{aligned} \tag{8.17}$$

Note that the state equations will remain unchanged when multiplying the costate with a positive constant α. The costate equations of motion will simply scale when multiplying the costate with α. As a result, the state equations will behave exactly the same over time, when multiplying the initial costate by a positive factor. We therefore sample the initial costate from the unit sphere, and directly use that costate.

The optimal Hamiltonian cannot always be set to zero by means of choosing a positive factor α. Therefore, if the initial sampled costate is such that this is not possible, we should disregard that sample. For a given costate direction, there is always at least one (possibly negative) α that sets the optimal Hamiltonian to zero, so the chance of picking an invalid costate is at most fifty percent. Given the relative computational cost of solving the constraint equation for α, disregarding at most half of the samples is acceptable.

8.3.3 Energy efficiency with input constraints

As the last class of optimal control problems we look at the combination of the previous two: energy optimal control with input constraints. This combination is useful in practice, as energy optimal control could demand larger torques than the actuators can supply.

The cost function is the same as for the energy optimal case, Equation 8.6. Finding the optimal cost function becomes a constrained minimization. Effectively, we solve:

$$\text{argmax}_{\tau} \frac{1}{2}\tau^{\top}\tau + \mu^{\top} M^{-1}(q)\tau \tag{8.18}$$

$$\text{s.t.} \quad |\tau_i| \leq \bar{\tau}_i \quad \forall i \tag{8.19}$$

The level sets of this function are hyperspheres, and the unconstrained minimum is at $\tau = -M^{-\top}\mu$. Since the constraints form a box, we can find a closed form solution:

$$\tau^* = \text{clip}(-M^{-\top}\mu, -\bar{\tau}, \bar{\tau}) \tag{8.20}$$

That is, clipping the unconstrained optimum to fall within the input bounds. The resulting optimal equations of motion are again derived by taking the appropriate derivatives.

To solve the constraint $\mathcal{H}^* = 0$ we again select a unit costate, and multiply it by an unknown positive factor α. Depending on α, the clipping will be turned off or on for different motor torques. To find the values of α at which the clipping behavior switches, we solve $\bar{\alpha} = \bar{\tau} \oslash (M^{-\top}(q)\hat{\mu})$.

The values in $\bar{\alpha}$ define intervals for the different clipping modes. For each interval, we fix the clipping behavior to solve $\mathcal{H} = 0$ for α. By fixing the clipping behavior, that constraint becomes either a quadratic equation, with a positive and a negative solution, similar to the unconstrained case, or the exact linear equation for the time optimal case. We check if the solutions (if they exist) fall in the domain for that clipping behavior. If so, we use that value for α.

Note that this approach only works if there is at most one solution, and that it exists most of the time.

Fortunately, the shape of the optimal Hamiltonian as function of α is not as complex as it might appear. Because for a given $\hat{\mu}$, the term $-M^{-\top}\hat{\mu}$ is a constant vector c , the Hamiltonian consists of the constant time-cost c_t, terms linear in α, and the sum over i of terms with the following structure:

$$\frac{1}{2}\text{clip}(\alpha c_i, -\frac{\bar{\tau}_i}{|c_i|}, \frac{\bar{\tau}_i}{|c_i|})^2 - \alpha c_i \text{clip}(\alpha c_i, -\frac{\bar{\tau}_i}{|c_i|}, \frac{\bar{\tau}_i}{|c_i|}) \tag{8.21}$$

where c_i are elements of the vector c, and $\bar{\tau}_i$ elements of $\bar{\tau}$. The equation above is quadratic within the clipping bounds, and linear outside them. Furthermore, the equation is smooth (with a non-smooth derivative). As a result, the Hamiltonian is a well behaved mix between energy optimal control without input bounds, and time optimal control with input bounds.

Therefore, the optimal Hamiltonian has a shape that has either one zero (if the linear term is larger than the linear parts of the clipped terms), or two zeros. In the latter case, there must be a zero for both a positive and a negative value of α. As a result, sampling from the sphere, and using the iterative procedure outlined above is a valid procedure to sample the costate for the problem of energy optimal control of input constrained systems.

8.4 | Conclusion

In this chapter we provided a proof of probabilistic completeness for RRT-CoLearn, and showed that there exists a simple costate sampling scheme for a large class of robotic systems and three important control problems. The code for the sampling scheme will be made available as a bitbucket repository.

The results are a step towards implementing RRT-CoLearn on multi degree-of-freedom systems. In future work, the machine learning aspects of RRT-CoLearn should be improved to make such an implementation feasible.

9

Discussion and Conclusion

9.1 | Discussion

This thesis was inspired by human movement to investigate three aspects of robotic arms: self-stability, energy-efficient actuation and trajectory planning. This chapter will first discuss the results for each of these aspects separately. Then, it will reflect upon the content of the thesis as a whole. In particular, I will look at the main commonality between all parts of this thesis: the use of optimal trajectory generation. The last two sections in this chapter will state the main conclusions and suggest directions for future work.

9.1.1 Self-stability

Overview of results

Chapter 2-5 researched the self-stability of industrial robot arms. This topic was addressed in three stages.

The first stage showed that a feedforward only controller can make robot arms exhibit a stable limit cycle. First, a simulation study found self-stabilizing reaching motions exist for a planar two degree of freedom arm, using an optimal control approach. Implemented on a physical robot, the motions were found to be self-stabilizing as well. However, the motions on the robot did not reach the desired end positions.

The second stage developed a robustly stable robot arm, such that a robot with limited sensing can learn to perform a pick-and-place task, despite both model uncertainty and disturbances. First, conditions were derived for which a motion is stable, independent on the input required to perform that motion. Furthermore, this stability is robust to uncertainties in the model. Using these conditions, a robustly stable limit cycle was found. Finally, the correct input to track that cycle was learned, using repetitive learning control. This scheme allows a robot to perform a pick-and-place task with feedforward control only, after a learning phase with a limited form of feedback. The feedback during the learning phase is delayed by one cycle of the reaching motion, which in effect means a delay in the order of seconds.

Finally, the third stage took a step back, and answered the following research question: is the use of self-stabilizing effects useful to improve the energy efficiency and the accuracy of robotic arms? This question was answered by studying how self-stabilizing motions affect the performance of a robot arm when it has low-gain feedback available. It was found that the choice of motion affects the accuracy of a robot with low-gain feedback. Specifically, the results showed that a generic smooth motion performed nearly as accurately as a motion optimized for self-stability.

The applicability of self-stabilization in robot arms

The research on self-stabilization of robotic arms was inspired by human self-stabilization, and the success achieved by implementation on (mostly) legged robots. It was noted that this self-stabilization was not yet used on industrial-type robot arms. Here I reflect on the use of self-stabilization for robot arms.

Chapters 2-4 showed that it is possible to apply self-stabilization on these arms, even in the presence of uncertainty. Further research [133], not included in this thesis, focused specifically on suppressing uncertainty by using smart trajectories. These results show that it is possible to achieve self-stabilizing motions on robot arms.

However, this does not mean that self-stabilizing effects should be the main focus for future robot arm control. First, the self-stabilizing effect is not large. The dynamical effects that cause the motion to be stable, do not cause fast convergence. While springs can be added to improve convergence (as was done in Chapter 4, and in [28]), this reduces versatility and the system still needs multiple cycles to return to its desired trajectory. Note that in previous robots showing self-stability, it was mostly caused by impacts, which have an immediate and strong effect on the behavior of the system. These impacts also allow for fast dissipation of energy. For the motions studied in this thesis dissipation of excess energy happens slowly, by a small friction term, and during phases where negative work is performed by the motors. Because the self-stabilizing effect is not large, it is not sensible to rely on it completely in an industrial setting.

Secondly, there is an issue with modeling accuracies. In [133], I explicitly attempted to minimize the sensitivity to such errors, and found that this can help, but cannot diminish the error in all cases.

Chapter 5 investigated a more moderate approach to self-stabilization: allowing it to aid other types of control. Optimizing for self-stability was found to improve the accuracy of robot arms. However, the difference between an optimized trajectory and more standard, relatively smooth trajectories was only significant when the feedback gains are small.

It is interesting to compare these results with studies on human motion optimization. Many researchers currently believe that humans implicitly use an optimization algorithm to plan their motions [10, 172]. Unfortunately, they do not agree on the cost function humans optimize for. The most commonly cited cost functions are energy use, variance in performance and smoothness (jerk). All these cost functions cause smooth trajectories. In my studies on robot motion, I also found such smooth trajectories to be the most useful.

The developed self-stabilization methods are useful for three special case scenarios. First, it can be useful when feedback is unreliable due to measurement noise [91],

feedback delays, or sensor failure. This is the likely reason that self-stability seems to be an import part of human movement control. Unreliable feedback might arise for robots in settings with high radiation. Second, it can be necessary when feedback is unavailable. Robot designers can be forced to omit sensors due to size, cost or energy constraints. This reason was the cause for investigating robustness of feedforward signals in works related to micro robotics [13]. The last type of situation for which self-stability is useful, is when the natural dynamics of the system happen to make self-stabilization highly effective. Such has been the case for robot walking and juggling. It is expected that a large impact or contact element to the motion can make self-stabilization more effective. Self stabilization in such contact-motions is an under-investigated area for industrial robot arms.

9.1.2 Energy efficiency and actuation

Overview of results

Chapter 6 was inspired by the energy efficiency of human movement. More precisely, it was inspired by the fact that human muscles perform negative and positive work via different (chemical) mechanisms. This inspiration caused us to investigate brakes as a dedicated mechanism for doing negative work, and their use in robot arms. Their use does seem counterintuitive, as the negative work they do is simply turned into heat, rather than captured in a flywheel or battery. However, the low efficiency of such capturing might not warrant the additional complexity and weight of a capturing mechanism.

In contrast, brakes have some attractive properties. One, because they can be mechanically simple, lightweight and cheap. Two, they function well without state estimation, or high bandwidth control loops. As such, they can be robust and precise.

The main contribution of this part of the thesis is to introduce the concept of a plugless robot arm. This robot arm performs a pick-and-place motion, where the pick location is located higher than the place location. As a result, the pick-and-place motion release the potential energy of the package. In principle, this energy can be used to power the robot arm, so it does not need additional energy supply. Designing and controlling such a plugless arm is a challenging, and realistic problem for testing efficient robot design and control.

In this thesis, an interesting property of the plugless arm problem is used: as the energy is already applied, it should be possible to control the arm by using only dissipative actions. This control concept was named dissipatively actuated control. By studying dissipatively actuated control, I investigated the limits of using brakes as actuators for robot arms.

Three dissipatively actuated controllers were designed and tested: a model based geometric controller, a model based receding horizon controller using Monte-Carlo

tree search and a reinforcement learning controller. The performance of all algorithms was similar, with comparatively little tweaking of the control algorithms. This is likely explained by the inherent robustness of braking actions: as they can only slow down the motion, they steer the robot to (stable) equillibria. Therefore wild, dangerous and unstable motions are difficult if not impossible to accomplish. These properties make dissipatively actuated manipulation an attractive testbed for online learning algorithms on physical robots.

Towards a plugless robot arm

The plugless arm was conceived as an ultimate challenge for energy efficient robotics; building a robot arm that needs no external energy input to perform its pick-and-place task, other than the gravitational energy of the pick objects themselves. With the foundations of dissipative control as presented in this thesis, the road is now clear to tackle the challenge of building a plugless arm in a future research project.

As the task is a given for most robots, the only way to approach energy efficiency is to reduce energy losses. The plugless arm is no different. The three main energy losses are the electric energy consumption for the control-boards, friction losses in joints and gears and the copper losses in the motor.

The energy loss due to computation can be reduced by designing simpler controllers that run at lower frequencies and require less computation. The gain in energy efficiency is mostly because such controllers can run on simpler electronics which require less overhead. Friction losses in joints and gears can be reduced in the design of the robot. Apart from higher quality manufacturing, the main benefit is obtained by using rolling contact. Sliding contact, including those in gears, can be avoided by using spindles or specially designed planetary gear stages.

The main avoidable source of energy loss are the copper losses in the motor. These can be reduced in three ways. One, improving the efficiency of the motor. This is mostly done by selecting the motor and gearbox combination that is most suitable. Two, reducing the torque demands for executing the task by picking a more effective motion. A famous example is to straighten the knee joint of the stance leg in walking, such that the links directly support the body weight without relying on motor torque.

The third way to reduce energy consumption is to transfer or store energy. Various mechanisms can relieve the actuators of some of their torque demands. Two important examples are: one, passive elements in parallel with the actuator, which can replace actuators for the energy neutral part of the torque demands; and two, multiarticular actuators, which can transfer energy between links and can avoid simultaneously performing positive and negative work in two different actuators.

Elastic and multiarticular actuators are a more conceptual improvement, and offer interesting possibilities for future research. By combining novel mechanisms for energy transfer and storage for multi degree of freedom arm with careful mechanical design, a plugless robot arm with a modest weight and drop (say 1 kg over 1 m) might become a feasible prospect.

9.1.3 Adaptive planning

Overview of results

Chapters 7-8 researched a new idea on planning trajectories with minimal computational power. The ability to plan motion is widely understood to be one of the most important areas in robotics. Various fields, from data based approaches, such as reinforcement learning to geometric control approaches, for instance those based on flatness, all aim at increasing the speed with which motion plans for robots are made. The most important contribution of this chapter is to propose a new algorithm: RRT-CoLearn.

The RRT-CoLearn algorithm, is grounded in the philosophy that a combination of various approaches can benefit from the strength of each. Optimal control techniques leverage the structure of the system to more quickly solve planning problems. However, these approaches require models of the obstacle boundaries, which are not always available. Sampling based approaches are well suited for situations. A sampling based planner will always rely on randomness to explore the motions, and is therefore at least somewhat limited in its computational efficiency. Finally, learning based approaches offer quick online performance, at the cost of an offline learning phase. For general feedback control of robotic systems, the problems of learning are not yet solved. Furthermore, the number of effective states of robotic systems moving around moving obstacles is overwhelming, and learning a highly accurate controller for such a large dimensional system will always require a prohibitive amount of samples.

All learning RRTs combine the efficient computation of optimal control with the online speedup caused by machine learning. The sampling based planner handles long-term planning, thereby drastically reducing the number of samples required for learning. Furthermore, the sampling-based planner can take care of obstacles in future work. Although such future implementations will finetune the roles of each of the algorithms, the general layout of the algorithm with an optimization, a learning, and a sampling based planner has shown great promise.

The work in this thesis investigated a new choice for the optimal control algorithm used in learning RRT. Specifically, so called indirect optimal control, based on the maximum principle of Pontryagin was used to create the RRT-CoLearn algorithm. This indirect optimal approach allows faster uniform sampling of optimal trajectories. Furthermore, the control is parametrized in the costate, a vector with the

same length as the state. Due to this smaller parametrization, the control input were learned for the first time. As a result, both the offline and online phases of the learning based RRT are sped up by a factor 10.

Applicability of RRT-CoLearn

In this thesis RRT-CoLearn is only applied to a pendulum swing up. This begs the question: how useful is the algorithm for more realistic scenarios, such as multi degree-of-freedom robot arms. This question is discussed in two steps. One, can RRT-CoLearn be applied on such an arm? And two, what are the benefits of applying RRT-CoLearn?

Before RRT-CoLearn can be applied to a multi-degree of freedom robot arm, one main challenge needs to be overcome. Two studies [116, 124] have attempted to directly use the CoLearn algorithm from Chapter 7, without successful planning as a result. Those studies identified poor learning performance in the steering function as the main culprit. The challenges of learning the costate where already explained in Chapter 7, and the simple dataset cleaning algorithm there was shown to not scale well to higher dimensions. However, there are avenues to explore in improving the learning performance. Chief among these is the use of generative models, such as variational autoencoders or generative adversarial networks. These models can potentially capture the more complex relationship between planning task and steering input.

Should the steering function learning problems be solved, the RRT will have access to an accurate cost and steering function. To I knowledge, this would be the first time that a kinodynamic RRT is given these functions for a task with a high dimensional state. As such, unknown challenges with RRTs in such a domain might be revealed.

If these challenges are fixed, RRT-CoLearn will be a single query kinodynamic planning algorithm delivering a feasible, but non-optimal, trajectory to the goal. The main quality of RRT-CoLearn is that it will be fast at exploring the state space of a robot in a cluttered environment. As a result RRT-CoLearn will be beneficial when the following four criteria are met:

- The motion needs to be analyzed in state space. That is, the dynamics of the movement matter for the success of its execution.
- The robot needs to move through an environment with obstacles.
- A feasible motion should be hard to find. This must be true to such an extent that the optimality of the motion is to no concern.
- The situation changes, such that similar situations are not encountered repeatedly.

Many scenarios that are currently seen as amenable to robotic automation do not meet all four criteria. RRT-CoLearn is therefore truly an algorithm for future industries, where faster movement in high-skill tasks are demanded. A simple, but relevant scenario would be bin picking from a moving bin. A robot with that skill would provide a performance benefit in warehouse with fast changing and irregularly shaped goods, such as seasonal fruits and vegetables.

9.1.4 General discussion

Applicability of trajectory optimization

This thesis relied heavily on the use of optimization on simulated models, in order to compute the motions of various robots. This was done because optimization provides a natural way to phrase my research questions as a mathematical problem, while leaning on previous work on optimization and simulation algorithms to find a solution. In this section I reflect on some issues that might interfere with the applicability of trajectory optimization for robotics.

The reason for this reflection is that optimized attributes of a simulation model do not always transfer well to the physical world. Partially, this can be directly attributed to the gap between model and reality. This gap was explored in Chapters 3-4. In those chapters, an algorithm was developed that takes into account the model-reality mismatch. Such algorithms are a wider area of research, for example see [171]. These algorithms provide performance guarantees given bounded disturbances. Another approach is to optimize the expected performance, given some probability distribution over the uncertainties or inaccuracies in the model, see [169]. A different solution to the same problem can be found in online adaptation, as explained in [157].

However, the model mismatch is not the most important issue with optimizing simulated systems. The main problem is more philosophical: often an optimal solution is not the goal. Rather, a solution that performs, without failing some constraint is the goal. Because many optimizations problems contain active constraints, the optimization algorithm searches for solutions near the constraint boundary. This means that many of the settings tried during the algorithm fail, and feasibility is only guaranteed after full convergence. However, the gains made during the final stages in the optimization, are often merely overfitting to the model, and are therefore at best not visible in the real setup, but likely even detrimental.

In chapter 5 this effect is clearly seen. The simulation results clearly overfitted on some aspect of the model around a controller frequency of 3.5 Hz. This caused a slight peak in the performance of the simulated system around that frequency. The real system on the other hand showed a distinct dip in performance, when the controller resulting from the optimization was applied. In fact, a commonsense, not optimized controller outperformed the controller that was optimized in simulation.

To counteract the overfitting problem, future work can look at the field of machine learning, where the problem is well studied. The most common solutions there are regularization and early stopping, see [50]. While both solutions seem to be detrimental by intuition, they cause an increase in performance in practice. For the machine learning case of an optimization overfitting sampled data, this can be theoretically explained by an improvement in the bias-variance trade off. A similar trade off likely exists when overfitting to a simulation model, and is worth investigating in that setting. Note that for early stopping, the optimization algorithm has to provide mostly feasible intermediate solutions. This can likely be accomplished by using forms of regularization that cause the solutions to move away from the constraint boundaries.

9.2 | Conclusions

This thesis studies the use of industrial robot arms by investigating three aspects inspired by human arm movement: self-stability, actuation and energy efficiency and adaptability by fast planning. The main results are

1. Self stability alone is sufficient to make a robot perform a basic pick-and-place task, even in the presence of disturbances and model inaccuracies.

2. While the choice of feedforward controller has a noticable effect on the accuracy of robot arms, optimizing that controller for self-stability is only useful in special cases when limited feedback gains are possible.

3. Three different controllers for a dissipatively actuated robot arm were developed. These controllers can potentially form the basis for a plugless robot arm, i.e., a pick-and-place robot arm that gets all its required energy from a height difference between the pick- and place-locations.

4. The use of indirect optimal control as part of a sampling based planning algorithm with a learned distance metric, allows learning of the steering function and decreases the time used in the offline training phase and causes a 10-fold speedup in the online planning.

9.3 | Future work

Earlier chapters in this thesis already gave suggestions for future research. Here I list the two most interesting directions that follow directly from this thesis, and briefly discuss a more general outlook.

9.3.1 Self stability in general robot arm movements

Situations encountered by robots outside the lab setting are more complex than those encountered by the self-stabilizing robots in this thesis. Before self-stabilizing effects can be utilized in the scenarios listed in Section 9.1.1, the results in this thesis should be extended towards such more complex situations. Most notably, three improvements should be made.

First, the self-stability of robot arm through contact should be studied. The assessment of stability as developed in this thesis is valid for movements with a set sequence of contacts. However, finding self-stabilizing motions becomes much harder when the search involves optional contact. A possible avenue towards solving this issue is to use soft contact constraints, as proposed in [115]. Furthermore, deviations from the limit cycle could cause a different sequence of contacts to occur, which should be accounted for in the stability assessment. By necessity, this stability assessment should therefore investigate the behavior after finite deviations from the nominal trajectory. That is, the assessment should investigate the basin of attraction of the cycle.

Second, the self-stability of non-cyclic motions should be further developed. First, because practical tasks for robot arms require a variety of motions. For instance, a pick and place task will likely have different pick and place locations in a practical setting. Second, the study of non-cyclic motions makes it possible to study the stability of the motion before learning of the feedforward signal has completed. In this thesis I have implicitly assumed that the unlearned motion is close enough to the desired motion that it also converges to a stable cycle. For more complex scenarios, particularly those involving contact, this might not hold true. To study self-stability of non-cyclic motions, work on the basin of attraction could again be useful, most notably via the funnel concept [171]. In this concept, the regions of attractions of small trajectories are forced to link together, thereby enforcing the attraction of a combined trajectory.

Third, the techniques for finding stabilizing feedforward controllers should be extened to robot arms with more degrees-of-freedom. Partly, this can be accomplished by using newly developed software that efficiently computes the desired derivatives of the dynamics equations [27]. However, due to the increased search space more extensive development is necessary. Conditions for the existence of self-stable motions could aid in more efficiently searching for such motions.

9.3.2 Generative models for RRT-CoLearn

In any learning RRT, a database of trajectories is used to learn a distance function and a steering function. Note that the forward dynamics of the system imply a mapping from initial state and steering input to final state. The steering function however, is a mapping from initial and final state to steering input, which is akin

to an inverse of the forward dynamics function. For general inputs, this inverse is multivalued.

The current solution, a cleaning algorithm, relies on the k-nearest-neighbor algorithm, which becomes less reliable for higher dimensions. The idea behind the cleaning algorithm is to delete all but a consistent branch of solutions in the multivalued inverse described above. The challenge is in identifying a consistent branch, which now causes the reliance on nearest-neighbor algorithms.

There exists an alternative approach: using generative models. Generative models are built to sample from the underlying distribution of some dataset. For RRT-CoLearn, this means that we can sample any of the possibilities for the steering input that connects the given initial and goal states. Because the sampling does not need to be from a consistent branch, we no longer have to explicitly identify these branches. Development of such generative models is therefore an interesting direction for future research.

9.3.3 Effective motions for future robots

The work in this thesis focussed on generating effective motions for future robot arms in simplified situations with a singular focus. Real-world robot arms do not have such a singular focus. As such, the meaning of effective motions for more general scenarios should be studied. In these scenarios, combining all aspects of state estimation and control should lead to a robust skill, as performed by the robot. Future work should study how the concepts in this thesis can enhance such skills by improving the role of feedforward control.

This work would extend of Chapter 5, in which the interplay between motion and low-gain feedback was studied. In that chapter, challenges with state estimation, and the reproducability of actions were neglected. For real robots, these issues can greatly reduce the effectiveness of feedback, causing the system to rely more on the feedforward controller. Furthermore, for the reaching task studied in that chapter, the prime objective is accuracy in all directions. For more general tasks, some types of errors may degrade performance very quickly, while performance is insensitive for other errors. This performance sensitivity is discussed in [158, 189] for throwing and walking tasks respectively.

To study the role of feedforward control in robot skills, the space of controllers needs to be explored effectively. Recent improvements in optimal control techniques and data-driven techniques such as MCTS and (deep) reinforcement learning have paved the way for such exploration. The challenge for future research lies in combining these data-driven techniques with the more model based concepts in this thesis.

References

[1] ABB-robotics (1974). *"Abb, historical milestones: Asea irb 6, 1974"*. https://new.abb.com/products/robotics/home/about-us/historical-milestones. Accessed: 2018-10-01, used with permission.

[2] Albu-Schäffer, A., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., and Hirzinger, G. (2007). *"The dlr lightweight robot: design and control concepts for robots in human environments"*. Industrial Robot: an international journal, 34(5), pp. 376–385.

[3] Albu-Schäffer, A., Ott, C., and Hirzinger, G. (2007). *"A unified passivity-based control framework for position, torque and impedance control of flexible joint robots"*. The international journal of robotics research, 26(1), pp. 23–39.

[4] Allen, R. and Pavone, M. (2016). *"A Real-Time Framework for Kinodynamic Planning with Application to Quadrotor Obstacle Avoidance"*. AIAA Guidance, Navigation, and Control Conference. San Diego, California, USA, pp. 5021–5028.

[5] Arai, H. and Tachi, S. (1991). *"Position control of manipulator with passive joints using dynamic coupling"*. Robotics and Automation, IEEE Transactions on, 7(4), pp. 528–534.

[6] Asadi, E., Hoyle, A., and Arzanpour, S. (2011). *"Design of a magnetorheological damper-based haptic interface for rehabilitation applications"*. Journal of Intelligent Material Systems and Structures, 22(11), pp. 1269–1277.

[7] Atkeson, C. (2017). *"Muscle: Two actuators in one"*. https://www.cs.cmu.edu/~cga/muscle/. Online: accessed December 2017.

[8] Balasubramanian, R. and Santos, V.J. (2014). *"The human hand as an inspiration for robot hand development"*, volume 95. Springer.

[9] Banaszuk, A. and Hauser, J. (1995). *"Feedback linearization of transverse dynamics for periodic orbits"*. Systems & control letters, 26(2), pp. 95–105.

[10] Bays, P.M. and Wolpert, D.M. (2007). *"Computational principles of sensorimotor control that minimize uncertainty and variability"*. The Journal of physiology, 578(2), pp. 387–396.

[11] Becker, A. and Bretl, T. (2012). *"Approximate steering of a unicycle under bounded model perturbation using ensemble control"*. IEEE Transactions on Robotics, 28(3), pp. 580–591.

[12] Becker, A., Felfoul, O., and Dupont, P.E. (2014). *"Simultaneously powering and controlling many actuators with a clinical mri scanner"*. Intelligent

Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pp. 2017–2023.

[13] Becker, A., Habibi, G., Werfel, J., Rubenstein, M., and McLurkin, J. (2013). *"Massive uniform manipulation: Controlling large populations of simple robots with a common input signal"*. Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pp. 520–527.

[14] Becker, A., Onyuksel, C., and Bretl, T. (2012). *"Feedback control of many differential-drive robots with uniform control inputs"*. Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 2256–2262.

[15] Berenson, D., Abbeel, P., and Goldberg, K. (2012). *"A robot path planning framework that learns from experience"*. IEEE Int. Conf. on Robotics and Automation.

[16] Bharatheesha, M., Caarls, W., Wolfslag, W., and Wisse, M. (2014). *"Distance metric approximation for state-space RRTs using supervised learning"*. "IEEE/RSJ Int. Conf. on Intelligent Robots and Systems.

[17] Bharatheesha, M., Caarls, W., Wolfslag, W.J., and Wisse, M. (2014). *"Distance metric approximation for state-space rrts using supervised learning"*. Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pp. 252–257.

[18] Bhounsule, P.A., Cortell, J., Grewal, A., Hendriksen, B., Karssen, J.D., Paul, C., and Ruina, A. (2014). *"Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge"*. The International Journal of Robotics Research, 33(10), pp. 1305–1321.

[19] Bischoff, R., Kurth, J., Schreiber, G., Koeppe, R., Albu-Schäffer, A., Beyer, A., Eiberger, O., Haddadin, S., Stemmer, A., Grunwald, G., et al. (2010). *"The kuka-dlr lightweight robot arm-a new reference platform for robotics research and manufacturing"*. Robotics (ISR), 2010 41st international symposium on and 2010 6th German conference on robotics (ROBOTIK), pp. 1–8.

[20] Boyd, S.P., Barratt, C.H., Boyd, S.P., and Boyd, S.P. (1991). *"Linear controller design: limits of performance"*. Prentice Hall Englewood Cliffs, NJ.

[21] Brown, E., Rodenberg, N., Amend, J., Mozeika, A., Steltz, E., Zakin, M.R., Lipson, H., and Jaeger, H.M. (2010). *"Universal robotic gripper based on the jamming of granular material"*. Proceedings of the National Academy of Sciences, 107(44), pp. 18809–18814.

[22] Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlf-

shagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). *"A survey of monte carlo tree search methods"*. IEEE Transactions on Computational Intelligence and AI in games, 4(1), pp. 1–43.

[23] Bullo, F. and Lewis, A. (2004). *"Geometric control of mechanical systems"*, volume 49. Springer.

[24] Burden, S., Revzen, S., and Sastry, S. (2011). *"Dimension reduction near periodic orbits of hybrid systems"*. Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on, pp. 6116–6121.

[25] Calanca, A., Muradore, R., and Fiorini, P. (2016). *"A review of algorithms for compliant control of stiff and fixed-compliance robots"*. IEEE/ASME Transactions on Mechatronics, 21(2), pp. 613–624.

[26] Caron, S., Pham, Q.C., and Nakamura, Y. (2017). *"Completeness of randomized kinodynamic planners with state-based steering"*. Robotics and Autonomous Systems, 89, pp. 85–94.

[27] Carpentier, J. and Mansard, N. (2018). *"Analytical derivatives of rigid body dynamics algorithms"*. Robotics: Science and Systems (RSS 2018).

[28] Chang, H., Kim, S.J., and Kim, J. (2016). *"Analytical investigation of the stabilizing function of the musculoskeletal system using lyapunov stability criteria and its robotic applications"*. Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pp. 5391–5398.

[29] Chaslot, G., Bakkes, S., Szita, I., and Spronck, P. (2008). *"Monte-carlo tree search: A new framework for game ai."* AIIDE.

[30] Chen, W., Yu, Y., Zhao, X., Zhao, L., and Sun, Q. (2011). *"Position control of a 2dof underactuated planar flexible manipulator"*. Mechatronics and Automation (ICMA), 2011 International Conference on, pp. 464–469.

[31] Cheng, G. and Wisse, M. (2017). *"Factory in a day, 7th framework programme project"*. http://www.factory-in-a-day.eu/. Accessed: December 2017.

[32] Chesi, G. (2011). *"Domain of attraction: Analysis and control via sos programming"*. Springer.

[33] Clary, P.J. and Hurst, J.W. (2017). *"Predictive planning based on reactive control"*. Dynamic Walking.

[34] Collins, S., Ruina, A., Tedrake, R., and Wisse, M. (2005). *"Efficient bipedal robots based on passive-dynamic walkers"*. Science, 307(5712), pp. 1082–1085.

[35] Collins, S.H., Wiggin, M.B., and Sawicki, G.S. (2015). *"Reducing the energy cost of human walking using an unpowered exoskeleton"*. Nature.

[36] Cordo, P., Carlton, L., Bevan, L., Carlton, M., and Kerr, G.K. (1994). *"Proprioceptive coordination of movement sequences: role of velocity and position information"*. Journal of Neurophysiology, 71(5), pp. 1848–1861.

[37] Dahiya, R.S., Metta, G., Valle, M., and Sandini, G. (2010). *"Tactile sensingâĂŤfrom humans to humanoids"*. IEEE transactions on robotics, 26(1), pp. 1–20.

[38] De Luca, A., Albu-Schaffer, A., Haddadin, S., and Hirzinger, G. (2006). *"Collision detection and safe reaction with the dlr-iii lightweight manipulator arm"*. Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pp. 1623–1630.

[39] De Schutter, J. and Van Brussel, H. (1988). *"Compliant robot motion i. a formalism for specifying compliant motion tasks"*. The International Journal of Robotics Research, 7(4), pp. 3–17.

[40] Debrouwere, F., Vukov, M., Quirynen, R., Diehl, M., and Swevers, J. (2014). *"Experimental validation of combined nonlinear optimal control and estimation of an overhead crane"*. IFAC Proceedings Volumes, 47(3), pp. 9617–9622.

[41] Dellon, B. and Matsuoka, Y. (2009). *"Modeling and system identification of a life-size brake-actuated manipulator"*. Robotics, IEEE Transactions on, 25(3), pp. 481–491.

[42] Desmurget, M. and Grafton, S. (2000). *"Forward modeling allows feedback control for fast reaching movements"*. Trends in cognitive sciences, 4(11), pp. 423–431.

[43] Deurzen, K. van (2016). *"Team delft doet mee aan amazon picking challenge"*. https://www.tudelft.nl/2016/tu-delft/team-delft-doet-mee-aan-amazon-picking-challenge. Accessed: 2018-10-01, used with permission.

[44] Diehl, M., Mombaur, K., and Noll, D. (2009). *"Stability optimization of hybrid periodic systems via a smooth criterion"*. Automatic Control, IEEE Transactions on, 54(8), pp. 1875–1880.

[45] Duindam, V., Macchelli, A., Stramigioli, S., and Bruyninckx, H. (2009). *"Modeling and control of complex physical systems: the port-hamiltonian approach"*. Springer Science & Business Media.

[46] Dupont, P.E. (1992). *"The effect of coulomb friction on the existence and uniqueness of the forward dynamics problem"*. Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on, pp. 1442–1447.

[47] Elbanhawi, M. and Simic, M. (2014). *"Sampling-based robot motion plan-*

ning: A review". Ieee access, 2, pp. 56–77.

[48] Erdmann, M.A. and Mason, M.T. (1988). *"An exploration of sensorless manipulation"*. IEEE Journal on Robotics and Automation, 4(4), pp. 369–379.

[49] Flash, T. and Hogan, N. (1985). *"The coordination of arm movements: an experimentally confirmed mathematical model"*. The journal of Neuroscience, 5(7), pp. 1688–1703.

[50] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *"The elements of statistical learning"*, volume 1. Springer series in statistics New York.

[51] Friston, K. (2010). *"The free-energy principle: a unified brain theory?"* Nature Reviews Neuroscience, 11(2), pp. 127–138.

[52] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. (2012). *"A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches"*. IEEE Transactions on Systems, Man, and Cybernetics, 42(4), pp. 463–484.

[53] Gantmacher, F.R. (1960). *"The theory of matrices"*, volume 2. Taylor & Francis US.

[54] Gao, D. and Book, W.J. (2010). *"Steerability in planar dissipative passive robots"*. The International Journal of Robotics Research, 29, pp. 353–366.

[55] Geraerts, R. and Overmars, M.H. (2004). *"A comparative study of probabilistic roadmap planners"*. Algorithmic Foundations of Robotics V, pp. 43–57.

[56] Glassman, E. and Tedrake, R. (2010). *"A Quadratic Regulator-Based Heuristic for Rapidly Exploring State Space"*. IEEE Int. Conf. on Robotics and Automation, pp. 5021–5028.

[57] Goodfellow, I. (2016). *"NIPS 2016 tutorial: Generative adversarial networks"*. arXiv preprint arXiv:1701.00160.

[58] Goretkin, G., Perez, A., Platt Jr, R., and Konidaris, G. (2013). *"Optimal sampling-based planning for linear-quadratic kinodynamic systems"*. IEEE Int. Conf. on Robotics and Automation.

[59] Goswami, A., Espiau, B., and Keramane, A. (1996). *"Limit cycles and their stability in a passive bipedal gait"*. Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on, volume 1, pp. 246–251.

[60] Goswami, A. and Peshkin, M. (1999). *"Mechanically implementable accommodation matrices for passive force control"*. The International Journal of

Robotics Research, 18(8), pp. 830–844.

[61] Goswami, A., Peshkin, M.A., and Colgate, J.E. (1990). *"Passive robotics: an exploration of mechanical computation"*. American Control Conference, 1990, pp. 2791–2796.

[62] Harris, C.M. and Wolpert, D.M. (1998). *"Signal-dependent noise determines motor planning"*. Nature, 394(6695), pp. 780–784.

[63] Hernández, C. (2017). *"Rosin, h2020 project"*. http://rosin-project.eu/. Accessed: December 2017.

[64] Hernandez, C., Bharatheesha, M., Ko, W., Gaiser, H., Tan, J., Deurzen, K. van, Vries, M. de, Van Mil, B., Egmond, J. van, Burger, R., et al. (2016). *"Team delft's robot winner of the amazon picking challenge 2016"*. arXiv preprint arXiv:1610.05514.

[65] Herzog, W. (2013). *"Mechanisms of enhanced force production in lengthening (eccentric) muscle contractions"*. Journal of Applied Physiology, 116(11), pp. 1407–1417.

[66] Hobbelen, D. and Wisse, M. (2007). *"Limit cycle walking"*. Humanoid Robots, Human-like Machines.

[67] Hobbelen, D.G. and Wisse, M. (2007). *"A disturbance rejection measure for limit cycle walkers: The gait sensitivity norm"*. IEEE Transactions on robotics, 23(6), pp. 1213–1224.

[68] Hogan, N. (1984). *"An organizing principle for a class of voluntary movements"*. The Journal of Neuroscience, 4(11), pp. 2745–2754.

[69] Holzinger, M., DiMatteo, J., Schwartz, J., and Milam, M. (2008). *"Passively safe receding horizon control for satellite proximity operations"*. Decision and Control, 2008. CDC 2008. 47th IEEE Conference on, pp. 3433–3440.

[70] Hsu, D., Kindel, R., Latombe, J.C., and Rock, S. (2002). *"Randomized Kinodynamic Motion Planning with Moving Obstacles"*. Int. J. Robotics Research.

[71] Hürmüzlü, Y. and Moskowitz, G. (1986). *"The role of impact in the stability of bipedal locomotion"*. Dynamics and Stability of Systems, 1(3), pp. 217–234.

[72] Hürmüzlü, Y. and Moskowitz, G. (1986). *"The role of impact in the stability of bipedal locomotion"*. Dynamics and Stability of Systems, 1(3), pp. 217–234.

[73] Jaillet, L., Cortés, J., and Siméon, T. (2010). *"Sampling-based path planning on configuration-space costmaps"*. IEEE Transactions on Robotics, 26(4), pp. 635–646.

[74] Junius, K., Moltedo, M., Cherelle, P., Rodriguez-Guerrero, C.D., Vander-

borght, B., and Lefeber, D. (2017). *"Biarticular elements as a contributor to energy efficiency: biomechanical review and application in bio-inspired robotics"*. Bioinspiration & Biomimetics.

[75] Karaman, S. and Frazzoli, E. (2011). *"Sampling-based Algorithms for Optimal Motion Planning"*. Int. J. Robotics Research, 30, pp. 846–894.

[76] Karaman, S. and Frazzoli, E. (2013). *"Sampling-Based Optimal Motion Planning for Non-holonomic Dynamical Systems"*. IEEE Int. Conf. on Robotics and Automation, pp. 5041–5047.

[77] Kavraki, L.E., Svestka, P., Latombe, J.C., and Overmars, M.H. (1996). *"Probabilistic roadmaps for path planning in high-dimensional configuration spaces"*. IEEE transactions on Robotics and Automation, 12(4), pp. 566–580.

[78] Kawato, M. (1999). *"Internal models for motor control and trajectory planning"*. Current Opinion in Neurobiology, 9(6), pp. 718 – 727.

[79] Khalil, H. (2002). *"Nonlinear systems"*. Prentice-Hall.

[80] Kim, C.H., Yonekura, K., Tsujino, H., and Sugano, S. (2011). *"Physical control of the rotation of a flexible objectâĂŤrope turning with a humanoid robot"*. Advanced robotics, 25(3-4), pp. 491–506.

[81] Klatzky, R.L. and Lederman, S.J. (1999). *"Tactile roughness perception with a rigid link interposed between skin and surface"*. Perception & psychophysics, 61(4), pp. 591–607.

[82] Knepper, R.A., Srinivasa, S.S., and Mason, M.T. (2012). *"Toward a deeper understanding of motion alternatives via an equivalence relation on local paths"*. Int. J. Robotics Research, 31(2), pp. 167–186.

[83] Ko, C.H., Young, K.Y., Huang, Y.C., and Agrawal, S.K. (2013). *"Active and passive control of walk-assist robot for outdoor guidance"*. Mechatronics, IEEE/ASME Transactions on, 18(3), pp. 1211–1220.

[84] Kocsis, L. and Szepesvári, C. (2006). *"Bandit based monte-carlo planning"*. European conference on machine learning, pp. 282–293.

[85] Koditschek, D.E. (1987). *"Adaptive techniques for mechanical systems"*. Fifth Yale Workshop on Applications of Adaptive Systems Theory, pp. 259–265.

[86] Koenemann, J., Del Prete, A., Tassa, Y., Todorov, E., Stasse, O., Bennewitz, M., and Mansard, N. (2015). *"Whole-body model-predictive control applied to the hrp-2 humanoid"*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015), p. 8p.

[87] Koolen, T., Bertrand, S., Thomas, G., De Boer, T., Wu, T., Smith, J.,

Englsberger, J., and Pratt, J. (2016). *"Design of a momentum-based control framework and application to the humanoid robot atlas"*. International Journal of Humanoid Robotics, 13(01), p. 1650007.

[88] Körding, K.P. and Wolpert, D.M. (2006). *"Bayesian decision theory in sensorimotor control"*. Trends in cognitive sciences, 10(7), pp. 319–326.

[89] Kragten, G.A. and Herder, J.L. (2010). *"The ability of underactuated hands to grasp and hold objects"*. Mechanism and Machine Theory, 45(3), pp. 408–425.

[90] Kuffner, J.J. and LaValle, S.M. (2000). *"Rrt-connect: An efficient approach to single-query path planning"*. IEEE Int. Conf. on Robotics and Automation, volume 2, pp. 995–1001.

[91] Kuo, A. (2002). *"The relative roles of feedforward and feedback in the control of rhythmic movements."* Motor control, 6(2), pp. 129–145.

[92] LaValle, S.M. (1998). *"Rapidly-exploring random trees: A new tool for path planning"*.

[93] LaValle, S.M. and Kuffner-Jr, J.J. (2001). *"Randomized Kinodynamic Planning"*. Int. J. Robotics Research, 20, pp. 378–400.

[94] Levine, S. and Koltun, V. (2013). *"Guided Policy Search"*. 30th Int. Conf. on Machine Learning.

[95] Li, Y., Littlefield, Z., and Bekris, K. (2016). *"Asymptotically optimal sampling-based kinodynamic planning"*. Int. J. Robotics Research.

[96] Linde, R. van der and Schwab, A. (1997). *"Lecture notes on multibody dynamics b, wb1413"*. Delft University of Technology.

[97] Lohmiller, W. and Slotine, J. (1999). *"Contraction analysis of nonlinear systems"*. Ph.D. thesis, Massachusetts Institute of Technology, Dept. of Mechanical Engineering.

[98] Longman, R.W. (2000). *"Iterative learning control and repetitive control for engineering practice"*. International Journal of Control, 73(10), pp. 930–954.

[99] López, C. and Martínez, E. (2000). *"Sub-finslerian metric associated to an optimal control system"*. SIAM Journal on Control and Optimization, 39(3), pp. 798–811.

[100] LÃűfberg, J. (2004). *"Yalmip : A toolbox for modeling and optimization in MATLAB"*. Proceedings of the CACSD Conference. URL http://users.isy.liu.se/johanl/yalmip.

[101] Majumdar, A. (2013). *"Robust online motion planning with reachable sets"*. Ph.D. thesis, Massachusetts Institute of Technology.

[102] Manchester, I. (2010). *"Transverse dynamics and regions of stability for nonlinear hybrid limit cycles"*. arXiv preprint arXiv:1010.2241.

[103] Manchester, I. and Slotine, J. (2012). *"Contraction criteria for existence, stability, and robustness of a limit cycle"*. arXiv preprint arXiv:1209.4433.

[104] Manchester, I.R., Tobenkin, M.M., Levashov, M., and Tedrake, R. (2011). *"Regions of attraction for hybrid limit cycles of walking robots"*. IFAC Proceedings Volumes, 44(1), pp. 5801–5806.

[105] Mansard, N. and Chaumette, F. (2007). *"Task sequencing for high-level sensor-based control"*. IEEE Transactions on Robotics, 23(1), pp. 60–72.

[106] Maschke, B., Ortega, R., and Van Der Schaft, A.J. (2000). *"Energy-based lyapunov functions for forced hamiltonian systems with dissipation"*. IEEE Transactions on automatic control, 45(8), pp. 1498–1502.

[107] Mathijssen, G., Lefeber, D., and Vanderborght, B. (2015). *"Variable recruitment of parallel elastic elements: Series–parallel elastic actuators (spea) with dephased mutilated gears"*. IEEE/ASME Transactions on Mechatronics, 20(2), pp. 594–602.

[108] Matsuoka, Y. and Townsend, B. (2001). *"Design of life-size haptic environments"*. Experimental Robotics VII, volume 271 of *Lecture Notes in Control and Information Sciences*, pp. 461–470.

[109] McGeer, T. et al. (1990). *"Passive dynamic walking"*. I. J. Robotic Res., 9(2), pp. 62–82.

[110] Milton, J.G. (2011). *"The delayed and noisy nervous system: implications for neural control"*. Journal of neural engineering, 8(6), p. 065005.

[111] Mitchell, R. (1972). *"Stability of the inverted pendulum subjected to almost periodic and stochastic base motionâĂŤan application of the method of averaging"*. International Journal of Non-Linear Mechanics, 7(1), pp. 101–123.

[112] Moerland, T.M., Broekens, J., and Jonker, C.M. (2017). *"Learning Multimodal Transition Dynamics for Model-Based Reinforcement Learning"*. arXiv preprint arXiv:1705.00470.

[113] Mombaur, K., Longman, R., Bock, H., and Schlöder, J. (2005). *"Open-loop stable running"*. Robotica, 23(1), pp. 21–33.

[114] Mombaur, K.D., Bock, H.G., Schlöder, J.P., and Longman, R.W. (2005). *"Open-loop stable solutions of periodic optimal control problems in robotics"*. ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik, 85(7), pp. 499–515.

[115] Mordatch, I., Popović, Z., and Todorov, E. (2012). *"Contact-invariant*

optimization for hand manipulation". Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation, pp. 137–144.

[116] Moring, S. (2018). *"Kinodynamic steering using supervised learning in rrt"*. M.sc. thesis, Delft University of Technology.

[117] Naidu, D.S. (2002). *"Optimal control systems"*, volume 2. CRC press.

[118] Neven, J. (2017). *"esign and evaluation of an energy-savig drive for a versatile robotic gripper"*. Master's thesis, Delft University of Technology.

[119] Oriolo, G. and Nakamura, Y. (1991). *"Control of mechanical systems with second-order nonholonomic constraints: Underactuated manipulators"*. Decision and Control, 1991., Proceedings of the 30th IEEE Conference on, pp. 2398–2403.

[120] Ortega, R. (1998). *"Passivity-based control of euler-lagrange systems: mechanical, electrical and electromechanical applications"*. Springer.

[121] Ortega, R., Perez, J.A.L., Nicklasson, P.J., and Sira-Ramirez, H.J. (2013). *"Passivity-based control of euler-lagrange systems: mechanical, electrical and electromechanical applications"*. Springer Science & Business Media.

[122] Palmieri, L. and Arras, K.O. (2015). *"Distance metric learning for rrt-based motion planning with constant-time inference"*. Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 637–643.

[123] Paluska, D. and Herr, H. (2006). *"The effect of series elasticity on actuator power and work output: Implications for robotic and prosthetic joint design"*. Robotics and Autonomous Systems, 54(8), pp. 667–673.

[124] Paramkusam, D. (2018). *"Comparison of optimal control techniques for learning-based rrt"*. M.sc. thesis, Delft Univsersity of Technology.

[125] Perez, A., Platt-Jr, R., Konidaris, G., Kaelbling, L., and Lozano-Perez, T. (2012). *"LQR-RRT*: Optimal Sampling-Based Motion Planning with Automatically Derived Extension Heuristics"*. IEEE Int. Conf. on Robotics and Automation.

[126] Pham, Q.C., Caron, S., Lertkultanon, P., and Nakamura, Y. (2016). *"Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots"*. Int. J. Robotics Research.

[127] Picasso, B. and Colaneri, P. (2008). *"A factorization approach for the l_∞-gain of discrete-time linear systems"*. Proc. of the 17-th IFAC world congress, Seoul, ROK, pp. 1299–1304.

[128] Plooij, M., De Vries, M., Wolfslag, W., and Wisse, M. (2013). *"Optimization of feedforward controllers to minimize sensitivity to model inaccuracies"*.

Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pp. 3382–3389.

[129] Plooij, M., Mathijssen, G., Cherelle, P., Lefeber, D., and Vanderborght, B. (2015). *"Lock your robot: A review of locking devices in robotics"*. Robotics Automation Magazine, IEEE, 22(1), pp. 106–117.

[130] Plooij, M., Vallery, H., and Wisse, M. (2016). *"Reducing the energy consumption of robots using the Bi-directional Clutched Parallel Elastic Actuator"*. IEEE Transactions on Robotics.

[131] Plooij, M. and Wisse, M. (2012). *"A novel spring mechanism to reduce energy consumption of robotic arms"*. Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 2901–2908.

[132] Plooij, M., Wolfslag, W., and Wisse, M. (2014). *"Open loop stable control in repetitive manipulation tasks"*. Robotics and Automation (ICRA), 2014 IEEE/RSJ International Conference on.

[133] Plooij, M., Wolfslag, W., and Wisse, M. (2015). *"Robust feedforward control of robotic arms with friction model uncertainty"*. Robotics and Autonomous Systems, 70(0), pp. 83 – 91.

[134] Plooij, M., Wolfslag, W., and Wisse, M. (2017). *"Clutched elastic actuators"*. IEEE/ASME Transactions on Mechatronics, 22(2), pp. 739–750.

[135] Pontryagin, L.S. (1987). *"Mathematical theory of optimal processes"*. CRC Press.

[136] Posa, M., Cantu, C., and Tedrake, R. (2014). *"A direct method for trajectory optimization of rigid bodies through contact"*. Int. J. Robotics Research, 33(1), pp. 69–81.

[137] Pratt, G.A. and Williamson, M.M. (1995). *"Series elastic actuators"*. Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on, volume 1, pp. 399–406.

[138] Pratt, J., Krupp, B., and Morse, C. (2002). *"Series elastic actuators for high fidelity force control"*. Industrial Robot: An International Journal, 29(3), pp. 234–241.

[139] Prilutsky, B.I. and Zatsiorsky, V.M. (1994). *"Tendon action of two-joint muscles: transfer of mechanical energy between joints during jumping, landing, and running"*. Journal of biomechanics, 27(1), pp. 25–34.

[140] Raibert, M.H. and Craig, J.J. (1981). *"Hybrid position/force control of manipulators"*. Journal of Dynamic Systems, Measurement, and Control,

103(2), pp. 126–133.

[141] Rao, A., Benson, D., Darby, C., Patterson, M., Francolin, C., Sanders, I., and Huntington, G. (2010). *"Algorithm 902: Gpops, a matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method"*. ACM Transactions on Mathematical Software, 37(2), pp. 1–39.

[142] Rao, A.V. (2009). *"A survey of numerical methods for optimal control"*. Advances in the Astronautical Sciences, 135(1), pp. 497–528.

[143] Reed, M.R. (2003). *"Development of an improved dissipative passive haptic display"*. Ph.D. thesis, Georgia Institute of Technology.

[144] Revzen, S. and Guckenheimer, J. (2012). *"Finding the dimension of slow dynamics in a rhythmic system"*. Journal of The Royal Society Interface, 9(70), pp. 957–971.

[145] Sastry, S. (2013). *"Nonlinear systems: analysis, stability, and control"*, volume 10. Springer Science & Business Media.

[146] Schaal, S. and Atkeson, C.G. (1993). *"Open loop stable control strategies for robot juggling"*. Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on, pp. 913–918.

[147] Schaft, A. van der (2006). *"Port-hamiltonian systems: an introductory survey"*. M. Sanz-Sole, J. Soria, J.L. Verona, and J. Verdura, editors, Proceedings of the International Congress of Mathematicians, volume III, Invited Lectures, pp. 1339–1365.

[148] Schaft, A. van der, Jeltsema, D., et al. (2014). *"Port-hamiltonian systems theory: An introductory overview"*. Foundations and Trends® in Systems and Control, 1(2-3), pp. 173–378.

[149] Schenau, G.J.V.I. (1989). *"From rotation to translation: Constraints on multi-joint movements and the unique action of bi-articular muscles"*. Human Movement Science, 8(4), pp. 301–337.

[150] Scherer, C. and Weiland, S. (2000). *"Linear matrix inequalities in control"*. Lecture Notes, Dutch Institute for Systems and Control, Delft, The Netherlands.

[151] Schmerling, E., Janson, L., and Pavone, M. (2015). *"Optimal sampling-based motion planning under differential constraints: the drift case with linear affine dynamics"*. IEEE Conf. on Decision and Control.

[152] Seyfarth, A., Geyer, H., and Herr, H. (2003). *"Swing-leg retraction: a simple control model for stable running"*. Journal of Experimental Biology, 206(15), pp. 2547–2555.

[153] Shiriaev, A., Freidovich, L., and Gusev, S. (2010). *"Transverse linearization for controlled mechanical systems with several passive degrees of freedom"*. Automatic Control, IEEE Transactions on, 55(4), pp. 893–906.

[154] Shkolnik, A., Walter, M., and Tedrake, R. (2009). *"Reachability-guided sampling for planning under differential constraints"*. Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, pp. 2859–2865.

[155] Singer, N.C. and Seering, W.P. (1990). *"Preshaping command inputs to reduce system vibration"*. Journal of Dynamic Systems, Measurement, and Control, 112, pp. 76–82.

[156] Singhose, W., Seering, W., and Singer, N. (1994). *"Residual vibration reduction using vector diagrams to generate shaped inputs"*. Journal of Mechanical Design, 116, pp. 654–659.

[157] Slotine, J.J.E., Li, W., et al. (1991). *"Applied nonlinear control"*, volume 199. Prentice hall Englewood Cliffs, NJ.

[158] Smeets, J., Frens, M., and Brenner, E. (2002). *"Throwing darts: timing is not the limiting factor"*. Experimental Brain Research, 144, pp. 268–274.

[159] Smith, H. and Blackburn, J.A. (1992). *"Experimental study of an inverted pendulum"*. American Journal of Physics, 60(10), pp. 909–911.

[160] Spaa, L. van der (2017). *"System dynamic design and control of the plugless robot arm: Towards energy neutral robotics"*.

[161] Stramigioli, S. (2001). *"Intrinsically passive control"*. Modeling and IPC control of interactive mechanical systemsâĂŤA coordinate-free approach, pp. 125–145.

[162] Strogatz, S. (2001). *"Nonlinear dynamics and chaos: with applications to physics, biology, chemistry and engineering"*.

[163] Sturm, J.F. (1999). *"Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones"*. Optimization Methods and Software, 11(1-4), pp. 625–653.

[164] Sutton, R.S. and Barto, A.G. (1998). *"Introduction to reinforcement learning"*, volume 135. MIT press Cambridge.

[165] Swanson, D. and Book, W. (2000). *"Torque feedback control of dry friction clutches for a dissipative passive haptic interface"*. Control Applications, 2000. Proceedings of the 2000 IEEE International Conference on, pp. 736–741.

[166] Tallavajhula, A., Choudhury, S., Scherer, S., and Kelly, A. (2016). *"List prediction applied to motion planning"*. IEEE Int. Conf. On Robotics and

Automation.

[167] Tassa, Y., Erez, T., and Todorov, E. (2012). *"Synthesis and stabilization of complex behaviors through online trajectory optimization"*. Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 4906–4913.

[168] Tedrake, R., Manchester, I.R., Tobenkin, M., and Roberts, J.W. (2010). *"Lqr-trees: Feedback motion planning via sums-of-squares verification"*. Int. J. Robotics Research, 29(8), pp. 1038–1052.

[169] Tempo, R., Bai, E.W., and Dabbene, F. (1997). *"Probabilistic robustness analysis: Explicit bounds for the minimum number of samples"*. Systems & Control Letters, 30(5), pp. 237–242.

[170] Thorpe, S., Fize, D., and Marlot, C. (1996). *"Speed of processing in the human visual system"*. nature, 381(6582), p. 520.

[171] Tobenkin, M., Manchester, I., and Tedrake, R. (2011). *"Invariant funnels around trajectories using sum-of-squares programming"*. Proceedings of the 18th IFAC World Congress, extended version available online: arXiv:1010.3013.

[172] Todorov, E. and Jordan, M.I. (2002). *"Optimal feedback control as a theory of motor coordination"*. Nature neuroscience, 5(11), pp. 1226–1235.

[173] Tonietti, G., Schiavi, R., and Bicchi, A. (2005). *"Design and control of a variable stiffness actuator for safe and fast physical human/robot interaction"*. Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, pp. 526–531.

[174] Ulrich, N. and Kumar, V. (1991). *"Passive mechanical gravity compensation for robot manipulators"*. Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on, pp. 1536–1541.

[175] Uno, Y., Kawato, M., and Suzuki, R. (1989). *"Formation and control of optimal trajectory in human multijoint arm movement"*. Biological cybernetics, 61(2), pp. 89–101.

[176] Van Nieuwstadt, M.J. and Murray, R.M. (1998). *"Real-time trajectory generation for differentially flat systems"*. International Journal of Robust and Nonlinear Control, 8(11), pp. 995–1020.

[177] Vermeulen, M. and Wisse, M. (2010). *"Intrinsically safe robot arm: Adjustable static balancing and low power actuation"*. International Journal of Social Robotics, 2(3), pp. 275–288.

[178] Webb, D.J. and Berg, J. van den (2013). *"Kinodynamic RRT*: Asymptot-*

ically Optimal Motion Planning for Robots with Linear Dynamics". IEEE Int. Conf. on Robotics and Automation.

[179] Wells, R. (1988). *"Mechanical energy costs of human movement: an approach to evaluating the transfer possibilities of two-joint muscles"*. Journal of biomechanics, 21(11), pp. 955–964.

[180] Westervelt, E., Grizzle, J., and Koditschek, D.E. (2003). *"Hybrid zero dynamics of planar biped walkers"*. Automatic Control, IEEE Transactions on, 48(1), pp. 42–56.

[181] Whitney, J.P. and Hodgins, J.K. (2014). *"A passively safe and gravity-counterbalanced anthropomorphic robot arm"*. Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 6168–6173.

[182] Wikimedia (2015). *"Industrial-robots"*. https://commons.wikimedia.org/wiki/File:Industrial-robots.jpg. Made by user: ISAPUT, Accessed: 2018-10-01.

[183] Winters, J.M. (1990). *"Hill-based muscle models: a systems engineering perspective"*. Multiple muscle systems, pp. 69–93.

[184] Wolfslag, W.J., Bharatheesha, M., Moerland, T.M., and Wisse, M. (2018). *"Rrt-colearn: towards kinodynamic planning without numerical trajectory optimization"*. IEEE Robotics and Automation Letters, 3(3), pp. 1655–1662.

[185] Wolfslag, W.J., Plooij, M., Babuska, R., and Wisse, M. (2015). *"Learning robustly stable open-loop motions for robotic manipulation"*. Robotics and Autonomous Systems, 66.

[186] Xu, Z. and Todorov, E. (2016). *"Design of a highly biomimetic anthropomorphic robotic hand towards artificial limb regeneration"*. Robotics and Automation (ICRA), 2016 IEEE International Conference on, pp. 3485–3492.

[187] Yang, Y., Merkt, W., Ivan, V., Li, Z., and Vijayakumar, S. (2017). *"Hdrm: A resolution complete dynamic roadmap for real-time motion planning in complex scenes"*.

[188] Yogeswaran, N., Dang, W., Navaraj, W.T., Shakthivel, D., Khan, S., Polat, E.O., Gupta, S., Heidari, H., Kaboli, M., Lorenzelli, L., et al. (2015). *"New materials and advances in making electronic skin for interactive robots"*. Advanced Robotics, 29(21), pp. 1359–1373.

[189] Zaytsev, P., Wolfslag, W., and Ruina, A. (2018). *"The boundaries of walking stability: viability and controllability of simple models"*. IEEE Transactions on Robotics, 34(2), pp. 336–352.

About the author

August 11, 1987
Born in Leiden, The Netherlands.

1998-2004
Gymnasium at Stedelijk Gymnasium Leiden. Profile: Natuur en Techniek and Natuur en Gezondheid

2005-2010
Bachelor of Science at Delft University of Technology. Final project: *Sensory weighting of force and position in motor control tasks in humans*

2010-2012
Master of Science at Delft University of Technology. Final project: *Reachability and task execution speed analysis of a resonating robotic arm.* Part of the Master education was an internship in the Robotics and BioMechancis Lab of Andy Ruina at Cornell University in the USA. The topic of the internship was *Controllability of the simplest walking model.*

2012-2016
PhD candidate at Delft University of Technology, of which this thesis presents the final results. The topic of the research was: *Effective robot arm motions*

2016-2018
Scientific researcher at Delft University of Technology, with a focus on learning state representations for robots in a reinforcement learning setting.

2018-present
Research associate at University of Edinburgh, with a focus on learning and planning recovery behaviours in quadruped locomotion.

List of publications

Journal papers

The boundaries of walking stability: viability and controllability of simple models
Petr Zaytsev, Wouter Wolfslag and Andy Ruina
Transactions on Robotics, Vol 34, 2017

RRT-CoLearn: towards kinodynamic planning without numerical trajectory optimization
Wouter Wolfslag, Mukunda Bharatheesha, Thomas Moerland and Martijn Wisse
Robotics and Automation Letters, Vol 3, 2018

A String-Based Representation and Crossover Operator for Evolutionary Design of Dynamical Mechanisms
Reinier Kuppens and Wouter Wolfslag
Robotics and Automation Letters, Vol 3, 2018

Clutched Elastic Actuators
Michiel Plooij, Wouter Wolfslag and Martijn Wisse
Transactions on Mechatronics, Vol 22, 2017

Robust feedforward control of robotic arms with friction model uncertainty
Michiel Plooij, Wouter Wolfslag and Martijn Wisse
Robotics and Autonomous Systems, Vol 70, 2015

Learning robustly stable open-loop motions for robotic manipulation
Wouter Wolfslag*, Michiel Plooij*, Robert Babuska and Martijn Wisse
Robotics and Autonomous Systems, Vol 66, 2015

Dissipatively actuated manipulation
Wouter Wolfslag, Michiel Plooij, Wouter Caarls, Sander van Weperen and Gabriel Lopes
Control Engineering Practice, Vol 34, 2015

Unparameterized optimization of the spring characteristic of parallel elastic actuators
Linda van der Spaa, Wouter Wolfslag, Martijn Wisse
Robotics and Automation Letters (under review)

* Shared first authorship

Conference papers

The effect of the choice of feedforward controllers on the accuracy of low gain controlled robots
Michiel Plooij*, Wouter Wolfslag* and Martijn Wisse
IEEE International Conference on Intelligent Robots and Systems, 2015

Open Loop Stable Control in Repetitive Manipulation Tasks
Michiel Plooij*, Wouter Wolfslag* and Martijn Wisse
IEEE International Conference on Robotics and Automation, 2014

Optimization of feedforward controllers to minimize sensitivity to model inaccuracies
Michiel Plooij, Michiel de Vries, Wouter Wolfslag and Martijn Wisse
IEEE International Conference on Intelligent Robots and Systems, 2013

Distance metric approximation for state-space RRTs using supervised learning
Mukunda Bharatheesha, Wouter Caarls, Wouter Wolfslag and Martijn Wisse
IEEE International Conference on Intelligent Robots and Systems, 2014

Generative CoLearn: steering and cost prediction with generative adversarial nets in kinodynamic RRT
Nick Tsutsunava, Wouter Wolfslag, Carlos Hernández Corbato and Martijn Wisse
IEEE International Conference on Robotics and Automation (under review)

Design and Evaluation of an Energy-Saving Drive for a Versatile Robotic Gripper
Job Neven, Mohamed Baioumy, Wouter Wolfslag and Martijn Wisse
IEEE International Conference on Robotic and Automation (under review)

Conference abstracts

Incorporating tasks in the dynamics of robotic arms
Wouter Wolfslag, Michiel Plooij, Irene Staal and Martijn Wisse
ICRA workshop on Task-based Optimal Design of Robots, 2013

Towards open loop stable manipulators
Wouter Wolfslag*, Michiel Plooij* and Martijn Wisse
Dynamic Walking Annual Meeting, 2013

* Shared first authorship

Acknowledgements

I would like to thank everyone who helped make my time as a PhD student a wonderful period. First up are my promotors: Martijn and Robert. I consider myself very lucky to have had you as guides. Specifically, I want to thank Robert for his sharp and precise feedback. I thank Martijn for giving me the opportunity of doing this research, and for his patience in teaching me how to write a little better. Most importantly, thank you for being a rolemodel for what I think people in academia should be. I also had two supervisors for work I did that was not directly related to this thesis: Heike and Jens. They have both inspired me with their clear analysis and discussion of the content of our work. I am hoping some of that has rubbed off on me.

I also did some supervising of my own. I thank Anne, Irene, Reinier, Evert, Ralph, Job, Linda, Koos, Moritz, Mohamed, Nick and the many others with whom I interacted more informally. I enjoyed tackling your theses with you, and learned a lot from your confused faces when I tried to explain my thoughts.

The main reason I found my time at the DBL so enjoyable were my colleagues. From scientific discussions to foosball, you were an intelligent and friendly bunch to hang out with, and were always ready to help out. Over the years there have been too many of you to list completely, so I will limit myself to the people I interacted with the most, my office-mates (Michiel, Daniël, Wietse, Shiqian, Daniel, Tim and Linda), the Bros and Brodettes (Mukunda, Joost, Jeff, Saher, Patricia and Carlos), and my co-authors not named before (Sander, Wouter, Gabriel and Thomas).

A special word goes out to my two paranymphs and friends, Mukunda and Michiel. Mukunda, working with you on RRT problems has been great fun. More imporantly, you were the perfect buddy for finishing up this thesis; your insights and enthusiasm provided great encouragement in those last few months. Michiel, sharing an office with you has been a privilage and a joy. I learned a lot from your insights in mechanisms and your approach for getting the work done. Mostly, I thank you for a plain good time: the terrible jokes and music that were the norm in our room, and the discussions on topics ranging from beer to philosophy, with a healthy dose of football and robotics sprinkled in.

Finally, I want to thank my friends and family for their support. Marte's suggestions turned the cover of this thesis into something I am very happy with. Mum and dad, thank you for always being there to listen to me and provide advice. Also, my thanks for hiding your exasperation when progress was slow, and for giving me the confidence to believe that all would be right. The very last words are for Hanneke, my girlfriend. Hanneke, your ability to select brilliantly between cheering me up and pushing me to do better (despite my protests) has made this thesis. With you, I look forward to the next adventure.

Propositions

1. Indirect optimal control is better at generating optimal trajectories for learning RRTs than direct optimal control. (This thesis)

2. Control using only dissipative actions is challenging from a planning perspective, but is beneficial because of robust real world results (This thesis).

3. A physical robot arm performing a pick-and-place task requires some minimum amount of feedback. (This thesis)

4. The role of feedforward control in robotics is currently undervalued because current robots have limited dynamical interaction with the environment.

5. The energetic cost of computation is unfairly neglected in models for optimizing energy efficiency of robots.

6. The utility of two equally effective methods is determined by the complexity of their implementation, not the simplicity of their concepts.

7. Advances in computation speed will never abolish the need for methods that require less computation.

8. The culture of judging robotic research by novelty undermines the value of that research.

9. The limited personal incentive for thorough reviewing is a danger to the reliability of science.

10. Car-navigation systems miss a vital option for setting a planning objective: plan in such a way, that the delays caused by missing exits is minimized.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotors Prof. dr. ir. M. Wisse and dr. ir. R.Babuška

Stellingen

1. Voor het genereren van optimale trajektoriën voor gebruik in RRTs is indirect optimal control beter geschikt dan direct optimal control. (Deze thesis)

2. Regelen met alleen maar dissipatieve acties zorgt voor een extra uitdaging bij het plannen van bewegingen, maar is nuttig door de robuste experimentele resultaten. (Deze thesis)

3. Een fysieke robotarm die een pick-and-place taak uitvoert heeft een minimale hoeveelheid feedback nodig. (Deze thesis)

4. De rol van feedforward control in robotica wordt onderschat, omdat de huidige robots slechts beperkte interactie met de omgeving hebben.

5. De energiekosten voor berekeningen worden onterecht verwaarloosd in modellen die gebruikt worden voor het optimaliseren van de energie efficiëntie van robots.

6. Het nut voor de gebruikers van twee net zo effectieve methoden wordt bepaald door de complexiteit van de uitvoering, niet door de eenvoud van de concepten.

7. Vooruitgang in rekenkracht van computers zal methodes die minder rekenkracht vereisen niet onnodig maken.

8. De cultuur die robotonderzoek beoordeelt op basis van nieuwigheid hindert de waarde van dat onderzoek.

9. Het gebrek aan persoonlijke prikkels tot het schrijven van grondige reviews is een gevaar voor de betrouwbaarheid van de wetenschap.

10. Navigatie systemen voor auto's missen een belangrijke planningsoptie: het dusdanig plannen dat vertragingen veroorzaakt door het missen van afslagen worden geminimaliseerd.

Deze stellingen worden opponeerbaar en verdedigbaar geacht en zijn als zodanig goedgek