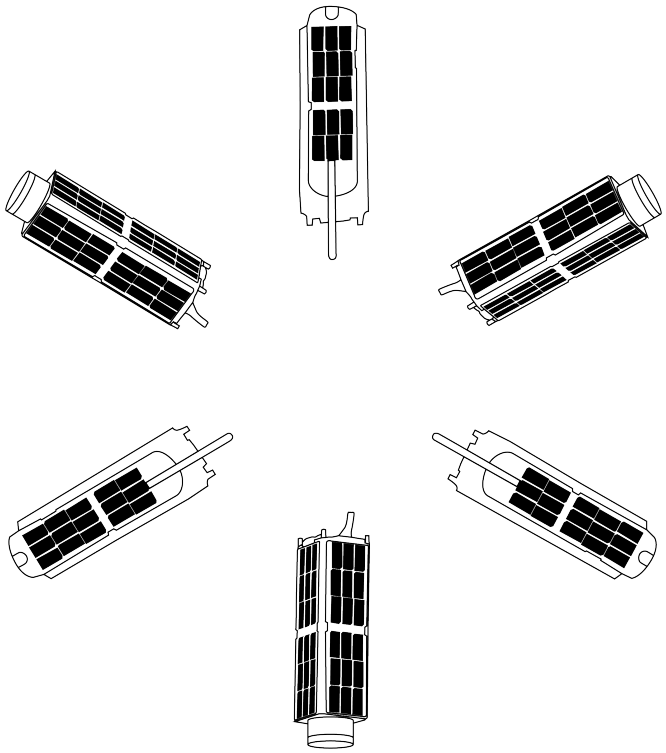
Distributed Collision Free Trajectory Optimization for the Reconfiguration of a Spacecraft Formation

Floris van Dam

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Distributed Collision Free Trajectory Optimization for the Reconfiguration of a Spacecraft Formation

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Floris van Dam

October 18, 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

In recent years there has been an increasing interest in formation flying with many lightweight spacecraft, as satellite missions can potentially become cheaper and more flexible. An example of such mission is the Silicon Wafer Integrated Femtosatellites mission, consisting of 100 to 1000 spacecraft with a mass of 0.1 kg. Due to modest control capabilities of the spacecraft and a higher risk of collisions, the requirements on trajectory optimization algorithms are increased. In this thesis a distributed trajectory optimization algorithm is developed which minimizes the required fuel for collision-free reconfiguration trajectories. First, the balance between cooperation in the formation and fuel consumption is investigated: with less cooperation the trajectory optimization algorithm can easily be distributed but the resulting fuel consumption is higher. The problem can also be distributed using dual methods, which can result in the same solution as a centralized algorithm. In literature both dual decomposition and the Alternating Direction Method of Multipliers (ADMM) in consensus form are proposed to solve this specific problem. In this thesis it is demonstrated that the Jacobian decomposition of the Augmented Lagrangian Method outperforms both dual decomposition and ADMM in terms of convergence rate. Furthermore it is shown that this algorithm does also converge in an asynchronous setting. Finally, the synchronous algorithms are significantly accelerated using Heavy Ball acceleration, the Fast Iterative Shrinkage-Threshold Algorithm and Anderson Acceleration.

Table of Contents

Acknowledgements **xi**

1 Introduction **1**
1-1 State of the Art . . . 3
1-2 Research Problem and Questions . . . 5
1-3 Main Contributions . . . 6
1-4 Thesis Outline . . . 7

2 Problem Formulation **9**
2-1 The Non-convex Trajectory Optimization Problem . . . 9
2-2 Initial, Final and Dynamical Constraint . . . 10
2-2-1 Reference Frames . . . 10
2-2-2 Perturbations in Low Earth Orbit . . . 11
2-2-3 Models in Literature . . . 12
2-2-4 The Xu-Wang Relative Dynamics Model . . . 12
2-2-5 Linearization and Discretization . . . 14
2-2-6 Passive Periodic Relative Orbits . . . 15
2-3 A Convex Collision Avoidance Constraint . . . 15
2-4 The Convex Problem with Sequential Convex Programming . . . 16
2-5 Simulation Specifics . . . 18
2-6 Summary and Conclusions . . . 19

3 Distributed Optimization with Decreased Cooperation **21**
3-1 Centralized algorithm . . . 21
3-2 Distributed Algorithm without Cooperation . . . 22
3-2-1 Model Predictive Control . . . 23
3-3 Hybrid Algorithm . . . 24
3-4 Simulations . . . 25
3-5 Summary and Conclusions . . . 29

4 Dual Methods **31**
4-1 Dual Decomposition . . . 31
4-1-1 Selecting an appropriate stepsize . . . 33
4-2 Augmented Lagrangian Methods . . . 40
4-2-1 Alternating Direction Method of Multipliers in Consensus Form . . . 40
4-2-2 Gauss-Seidel Decomposition of Augmented Lagrangian Method . . . 42
4-2-3 Jacobian Decomposition of Augmented Lagrangian Method . . . 47
4-2-4 Simulation and Comparison . . . 48
4-3 Asynchronous Algorithm . . . 50
4-4 Summary and Conclusions . . . 53

5 Accelerated Dual Methods with Sequential Convex Programming **55**
5-1 Accelerating Dual Algorithms . . . 55
5-1-1 Heavy Ball Acceleration . . . 55
5-1-2 Fast Iterative Shrinkage-Threshold Algorithm . . . 58
5-1-3 Anderson Acceleration . . . 61
5-1-4 Summarizing Acceleration Figures . . . 63
5-2 Simulations with Sequential Convex Programming . . . 66
5-2-1 Four Spacecraft Reconfiguration . . . 66
5-2-2 Ten Spacecraft Reconfiguration . . . 66
5-2-3 Computation Times . . . 66
5-3 Summary and Conclusions . . . 70

6 Conclusions **71**
6-1 Summary of the Research and Answers to Sub-Questions . . . 71
6-2 Conclusion of the Research . . . 73
6-3 Recommended Future Work . . . 74

Bibliography **77**

Glossary **81**
List of Acronyms . . . 81
List of Symbols . . . 82

List of Figures

1-1 The difference between a constellation, formation and swarm is characterized by the required control accuracy and inter-satellite distance. Adopted and adjusted from [1]. 2

2-1 Visualization of the Earth Centered Initial (ECI) frame $(\hat{X}, \hat{Y}, \hat{Z})$ and the Local-Vertical-Local-Horizontal (LVLH) frame $(\hat{x}, \hat{y}, \hat{z})$, adapted and adjusted from [2]. 11

3-1 Relative distances between the spacecraft pairs before and after collisions are avoided. When the constraint is active, all relative trajectories are pushed above R_{col}. 26

3-2 Acceleration magnitudes corresponding to Figure 3-1. As each spacecraft (Sc) has 6 DOF control, the magnitudes of the accelerations in all DOF are summed per sampling instance. 27

3-3 Obtained acceleration magnitudes when $(2\text{-}14)$ is solved with objective $\sum_i^N \|U_i\|_1$ instead of $\sum_i^N \frac{1}{2} U_i^T U_i$. This results in sparse acceleration vectors. 28

4-1 Dual decomposition with diminishing stepsize $\rho = \frac{5\times10^{-5}}{k}$ versus constant stepsize $\rho = 3 \times 10^{-5}$. 34

4-2 Lagrangian function value in stable versus unstable dual decomposition. 38

4-3 Unstable dual decomposition, relative distances between spacecraft pairs in iteration 6 versus iteration 7. 39

4-4 Two instances of the scaled Lagrangian function value over the iterations. This function value changes twice per iteration: after the u-update and after the y-update. Close to convergence, the u-update in which the Lagrangian should be minimized does not always decrease the Lagrangian function value. 39

4-5 Consensus ADMM for 4 spacecraft with different stepsizes ρ. 42

4-6 Convergence of GS-ALM with 4 spacecraft for different stepsizes ρ. 44

4-7 Convergence of all residual elements: most of them are negative and thus collision-free, only few are positive. Top figure: when ρ is chosen too large, different residual elements become largest over the iterations. Lower figure: when ρ is chosen small enough, the largest residual element converges almost monotonically. 45

4-8 The average required velocity change over the iterations for $\rho = 5 \times 10^{-3}$ versus $\rho = 7 \times 10^{-4}$. Note the y-scales: the top figure is much closer to $\Delta\bar{V}^*$ than the lower figure. 46

4-9 Convergence of GS-ALM: 4 spacecraft with random spacecraft update order. The properly chosen stepsize still results in convergence, whereas the stepsize chosen too large results in oscillations due to overcompensation of the spacecraft which updates first. 46

4-10 Convergence of J-ALM with 4 spacecraft for different stepsizes ρ and relaxation parameters ν. The red line with $\rho = 7 \times 10^{-4}$ and $\nu = 0.5$ converges fastest, although the primal residual is zero between iteration 145 and 235. Decreasing ν to 0.2 results in much slower convergence, increasing it to 0.7 results in undesired oscillations. 48

4-11 Asynchronous J-ALM, waiting for 2 control vector updates per iteration, random delay with $T_{\text{max}} = 5$ and $\nu = 0.5$. The stepsize suitable for synchronous J-ALM results in oscillations in the asynchronous case, but dividing the stepsize through the maximum delay results in convergence. 52

4-12 Asynchronous J-ALM, waiting for 2 control vector updates per iteration, random delay with $T_{\text{max}} = 15$, $\rho = (7/15) \times 10^{-4}$ and $\nu = 0.5$. Also for larger delays convergence can be obtained when the stepsize is adjusted appropriately. 52

5-1 Heavy Ball on dual decomposition, with $\rho = 3 \times 10^{-5}$ and different values for momentum parameter β. 56

5-2 Heavy Ball on synchronous J-ALM with $\rho = 7 \times 10^{-4}$, $\nu = 0.5$ and $\beta = 0.5$. Updating U in the same way as dual vector y has a large influence on the convergence rate. 57

5-3 Heavy Ball on asynchronous J-ALM with $\rho = 1.4 \times 10^{-4}$, $\nu = 0.5$, $T_{\text{max}} = 5$ and a central node waiting for 2 control vectors. Momentum parameter $\beta = 0.1$ results in little acceleration whereas $\beta = 0.5$ does not converge. The algorithm can be more significantly accelerated with $\beta = 0.3$. 58

5-4 The two forms of FISTA in comparison with standard dual decomposition. Standard and FISTA-CD are simulated with $\rho = 3 \times 10^{-5}$; for FISTA-BT the largest converging stepsize was $\rho = 1 \times 10^{-7}$. FISTA-CD can significantly accelerate convergence. 59

5-5 Synchronous J-ALM with different implementations of FISTA. All algorithms are simulated with $\rho = 7 \times 10^{-4}$ and $\nu = 0.5$. When U is updated using the same momentum rule as y, the algorithm diverges; only FISTA-CD on y converges. . . 60

5-6 Synchronous J-ALM compared to FISTA-CD. When ρ is lowered from $\rho = 7 \times 10^{-4}$ to $\rho = 3 \times 10^{-4}$, the convergence rate increases. 60

5-7 Anderson Acceleration on dual decomposition with different values of m_k. All simulations are performed with $\rho = 3 \times 10^{-5}$. 62

5-8 Anderson Acceleration on synchronous J-ALM with $\rho = 4 \times 10^{-4}$; again the convergence rate is largest when U is updated using a similar rule as y. For this setup, $m_k = 2$ and $m_k = 5$ perform nearly similar. 62

5-9 Standard versus Anderson Accelerated asynchronous J-ALM with $\rho = 1.4 \times 10^{-4}$; when a lower bound on a is included, Anderson Acceleration can accelerate convergence. 63

5-10 Regular dual decomposition compared with different acceleration algorithms. All algorithms are simulated with $\rho = 3 \times 10^{-5}$, Anderson Acceleration clearly outperforms the other acceleration methods. 64

5-11 Standard synchronous J-ALM compared with three different acceleration algorithms, all with $\rho = 4 \times 10^{-4}$ and $\nu = 0.5$. 65
5-12 Standard asynchronous J-ALM compared with Heavy Ball and Anderson Acceleration, all with $\rho = 1.4 \times 10^{-4}$ and $\nu = 0.5$. 65
5-13 Four spacecraft: AA on dual decomposition with SCP. The algorithm converges consistently despite the updated collision avoidance constraint in each SCP iteration. Convergence in 7 SCP iterations. 67
5-14 Four spacecraft: AA on synchronous J-ALM with SCP. Convergence in 7 SCP iterations. 68
5-15 Ten spacecraft: AA on dual decomposition with SCP. Convergence in 4 SCP iterations. 69
5-16 Ten spacecraft: AA on synchronous J-ALM with SCP. Convergence in 4 SCP iterations. 69

List of Tables

3-1 Four spacecraft reconfiguration: Averaged convergence characteristics over 10 runs with $\sigma = 0.1$ km. The computation time is given per spacecraft per iteration. . . 26

3-2 Ten spacecraft reconfiguration: Averaged convergence characteristics over 10 runs with $\sigma = 0.3$ km. Computation time is given per spacecraft per iteration. 28

3-3 Convergence characteristics for 100 spacecraft formation. 28

4-1 Convergence characteristics for the three different algorithms on 4 spacecraft reconfigurations averaged over 3 simulations. 49

5-1 Convergence characteristics of Anderson Acceleration on dual algorithms in SCP. 67

5-2 Time to solve one dual iterations and to solve one SCP iteration for the two accelerated dual methods. 68

Acknowledgements

This report is written as graduation thesis for the master Systems and Control at the Delft University of Technology.

I would like to thank my supervisors, dr.ir. T. Keviczky for his guidance when in doubt about the next steps in this research, and Dr.ir. R. Fónod for his accurate and detailed fault detection and quality control. Furthermore, I am grateful to Stein Stroobants for discussing all bumps in the research road last year, and to Anna Kay Mastenbroek for an endless amount of support and good coffee. Finally, special thanks to my parents, who always keep faith in me.

Delft, University of Technology
October 18, 2019

Floris van Dam

"The first principle is that you must not fool yourself - and you are the easiest person to fool."

— *Richard P. Feynman*

Chapter 1

Introduction

In this research a distributed trajectory optimization algorithm is developed which minimizes the total fuel consumption of collision-free reconfiguration trajectories. This algorithm is especially relevant for formations with a large number of lightweight spacecraft with modest hardware capabilities.

Lightweight satellites have been around already since the sixties, with OSCAR-1, a simple 10 kg radiotransmitter launched to Low Earth Orbit (LEO), and the ECHO-I Balloon satellite, a 60 kg satellite being the world's first communication satellite [1]. However, these were passive spacecraft, without any means of control. The development of functionalities on small satellites has skyrocketed since 1999, the year in which the CubeSat design was proposed. This is a standardized platform of (possibly) multiple blocks with sides of 10 cm and a mass of maximum 1.33 kg connected to each other [3]. At first this platform was used for educational purposes on universities, but nowadays also government and industry is involved. At present all parties have developed and launched fully controllable nanosatellites [4]. Satellites are classified as nanosatellites when they have a weight between 1 and 10 kg. With a mass between 0.1 and 1 kg they are called picosatellites [5].

The Delft University of Technology is making its own contribution to this trend with the development of the Delfi-PocketQube (Delfi-PQ) picosatellite [6]. This project aims to develop a platform based on connected cubes with sides of 5 cm. The platform secures basic functionalities, but can easily be outfitted with more advanced payloads and subsystems by any interested party. As such lightweight spacecraft have only limited hardware capabilities, the strength comes mainly from its numbers. An example of an application with a large number of lightweight spacecraft is Planet Labs, a company with over 150 nanosatellites in a constellation with the goal of imaging the Earth. As they have so many spacecraft, they are able to monitor most of the places on Earth twice a day with a resolution of under 4 meters [7]. In the Silicon Wafer Integrated Femtosatellites (SWIFT) mission, hundreds of 100 gram spacecraft are proposed in a swarm architecture. Possible applications for such a mission are massive distributed sensor networks and atmospheric sampling. In Figure 1-1 the difference between a constellation, formation and swarm is visualized. Most of the launched and proposed missions with a large number of spacecraft are constellation missions. As the

inter-satellite distance is large, the risk of collisions is low. The required positions of spacecraft in a constellation are defined with respect to Earth, which means that each spacecraft is responsible for tracking a pre-determined orbit. In general the required control accuracy is low. In swarms, the trajectories of a large group of spacecraft can be defined either with respect to Earth, or to each other. The required accuracy of the relative distances is still not so important; as long as the swarm covers a certain area or volume and the spacecraft in the swarm do not collide. In formation flying missions on the other hand, most spacecraft track a trajectory which is defined with respect to other spacecraft instead of to Earth. Accurate control is required to obtain and maintain these relative states. A possible application for formation flying missions with a large number of spacecraft is space-based interferometry: instead of one large telescope, many smaller telescopes attached to spacecraft can be used to make an image of for example a part of the solar system. When the smaller telescopes are exactly at the right position at a specific time, the resulting images can be combined in an image with a far higher resolution than the image from the large telescope. Examples of such missions are MAXIM [8] and the Stellar Imager mission [9].

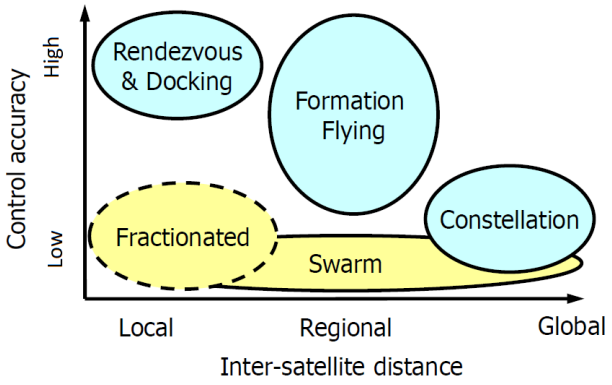


Figure 1-1: The difference between a constellation, formation and swarm is characterized by the required control accuracy and inter-satellite distance. Adopted and adjusted from [1].

Formation flying with many smaller spacecraft is more difficult than formation flying with few larger spacecraft due to four challenges: a high degree of autonomy in the formation is required, nano- and picosatellites can carry only little fuel, a high control accuracy is required despite modest hardware capabilities and the formation configuration has to be flexible. These challenges will be first be discussed more elaborately.

First, optimal collision-free trajectories cannot be computed by a centralized unit anymore. As the collision avoidance constraint is defined between each pair of spacecraft, the number of constraints increases quadratically with the number of spacecraft. To obtain flexibility and safety also in off-nominal situations, the spacecraft should be able to optimize their own trajectories without the need of a central processor.

Second, the reconfiguration and collision avoidance challenges have to be solved while minimizing the fuel consumption of the spacecraft. Whereas the spacecraft can be provided with solar panels to load batteries, the on-board propellant is limited and thus a limiting factor for the life-time of nano- and picosatellites in the formation.

Third, in formation flying missions a higher control accuracy is required than in swarms and constellations. The state-of-the-art in terms of control accuracy was demonstrated in

the Canadian Advanced Nanospace eXperiment-4&5 (CanX-4,5) mission which launched in 2014. Two 7 kg nanosatellites demonstrated four different configurations with a varying separation distance from 1 km to 50 meter. The two spacecraft were able to obtain a relative position control accuracy of less than one meter [10]. This means that the two spacecraft obtained a dynamical range, i.e. the inter-satellite distance divided by the control accuracy, of 10^3. This dimensionless quantity can be used to indicate the complexity of formation flying control. Future missions such as the MAXIM or Stellar Imager mission require a much higher dynamical range of 10^8 [11]. If the separation distance is for example 10 km, a control accuracy of 0.1 mm is required. This is far more accurate than present day accomplishments. Note that to obtain a high control accuracy also the state estimation should be very accurate. This is another challenge for nano- and picosatellites in a formation, but outside the scope of this research.

As a fourth challenge, formation flying missions require more flexible formation configurations. Whereas a spacecraft in a constellation typically remains in the same orbit after it is brought there, a formation might need to resize or rotate dependent on the mission objective [12], collisions have to be avoided within the formation and it has to adjust when spacecraft are added or subtracted from the formation. Both designing a reconfiguration maneuver and communicating the plans through the formation is a challenge for larger formations.

This research addresses the first two challenges: solving the collision-free trajectory optimization problem in a distributed way while minimizing the total fuel consumption. In this introduction chapter, first the state of the art of distributed trajectory optimization and asynchronous optimization is discussed. This is followed by the research problem and questions. Third, the main contributions of this research are summarized, and finally the outline of the entire thesis is given.

1-1 State of the Art

Before the research questions are defined, a small overview is given of the state of the art of distributed trajectory optimization and asynchronous optimization. Here distributed optimization is defined as an optimization problem in which multiple agents cooperate to minimize a global objective, without the need of a central agent. Distributing the spacecraft trajectory optimization problem is typically done by either decreasing the cooperation or by solving an equivalent dual problem [13].

One way to relax the problem formulation is to relax the cooperation, i.e. the responsibility for avoiding collisions. In a centralized algorithm, if two spacecraft are on track to collide both of them are responsible for avoiding the collision while minimizing the combined fuel consumption. The spacecraft thus fully cooperate to avoid the collision. Removing the cooperation is demonstrated for example in [2]. Here Morgan et al. assign varying collision avoidance responsibility to the spacecraft. Now only one spacecraft per pair is responsible to avoid a collision. All spacecraft minimize only their own fuel cost which will be relatively low for some spacecraft, but higher for the spacecraft which have to avoid many neighboring spacecraft. The trajectory optimization problem is cast in a Sequential Convex Programming (SCP) framework, where the iteration at which a trajectory converges depends on the collision-avoidance responsibility of the spacecraft. Due to the removal of cooperation, the

distributed solution may be further away from optimal than a centralized solution. A quantitative comparison of fuel consumption in fully cooperating versus non cooperating algorithms is not yet described in literature.

Dual methods are a popular choice for distributed optimization as they are able to obtain exactly the same solution as a centralized algorithm. Instead of increasing the cost, the iterative dual methods increase the time required to solve the problem. Different dual methods such as dual decomposition and Alternating Direction Method of Multipliers (ADMM) are often used for distributed optimization, with applications for example in power systems [14] and large scale image processing [15, 16]. Dual decomposition is applied to the spacecraft trajectory optimization problem in [17], where it is shown that the algorithm iteratively converges to the same collision-free trajectories as a centralized solution. Dual decomposition methods are however known for their slow convergence and the convergence rate is sensitive to parameter selection.

Many authors have applied the ADMM to the trajectory optimization problem, not only for spacecraft but also for robots [18], vehicles [19] or general agents in a network [20]. ADMM can be seen as a robust version of dual decomposition. The problem is decoupled as each agent solves the optimization problem using fixed estimates of the trajectories of other agents. Again in an iterative way, consensus is obtained between the estimates and the actual control vectors, eventually resulting in collision-free trajectories. ADMM is generally known to converge fast to modest accuracy, but very slow to high accuracy [21]. Instead of the consensus form of ADMM which uses three sets of variables, i.e. the control acceleration vectors, the estimates of trajectories and the dual vector, the algorithm can also be extended to more sets of variables. The control vector of each agent is then defined as a separate set of variables, resulting in the sequential update of each control vector followed by the dual update [22]. This algorithm, which is called the Gauss-Seidel decomposition of the Augmented Lagrangian Method (GS-ALM), is not yet implemented on the trajectory optimization problem. In a separate paper, the authors show that a slightly modified algorithm can also converge when all agents update in parallel, called Jacobian decomposition of the Augmented Lagrangian Method (J-ALM) [16]. This algorithm is demonstrated to converge in a shorter time than a commercial state-of-the-art optimization solver.

When problems and datasets become larger and distributed over multiple sites, which is the case for example in the machine learning field, the need for new asynchronous distributed optimization algorithms and convergence results increases. One possibility to implement asynchronous distributed optimization is by using an asynchronous incremental aggregated gradient method, as proposed in [23]. When the objective function is strongly convex and possibly constrained by convex constraints, a linear convergence rate is proven. Dual decomposition can also be implemented in an asynchronous way, as is shown in [24]. Instead of one agent updating the dual vector when all gradients are collected, the dual vector is updated with part of the gradients being just collected and accurate, and the other gradients stored in memory from a previous iteration and thus outdated. When an analytic expression is available for the dual gradient, it is shown that the convergence rate can be expressed analytically as well.

As all dual methods use a subgradient method to update the dual vector, various acceleration methods can be used to accelerate convergence. Well known methods are Heavy Ball Acceleration (HBA) [25] and a Nesterov type of acceleration called Fast Iterative Shrinkage-Threshold

Algorithm (FISTA) [26, 27], which use memory of the previous iterate to accelerate the update of the present iterate. A more advanced acceleration method is Anderson Acceleration (AA), which is recently also applied to constrained gradient methods [28].

1-2 Research Problem and Questions

This research aims to improve the state of the art of distributed trajectory optimization algorithms for finite-time collision-free reconfigurations. Different distributed algorithms will be compared on their ability to minimize the fuel consumption and on their convergence rate. To limit the scope of the research, some assumptions are introduced.

Assumptions:

- The initial and desired final states are given for and known to each spacecraft;
- All spacecraft have exact knowledge of their relative state with respect to the formation center;
- The reconfigurations discussed in this research take place in LEO;
- The spacecraft are homogeneous picosatellites, and all spacecraft have a specific thruster configuration capable of providing 3 Degree of Freedom (DOF) accelerations;
- The spacecraft have unlimited propellant onboard, they can perform arbitrary large reconfigurations;
- Attitude control is assumed to be perfect and instantaneous so that the 3 DOF acceleration control is in the direction of the axes of the Local-Vertical-Local-Horizontal (LVLH) frame;
- All spacecraft have an omni-directional antenna which has a specified maximum communication range. This enables spacecraft to communicate with each other when they are within this communication distance from each other;
- No communication delay or data loss is present.

Under these assumptions, this research formulates an answer to the following question:

How to design a distributed collision-free trajectory optimization algorithm for a large formation of spacecraft which finds the minimum-fuel solution in finite time?

This research question implies that the developed algorithm has some constraints:

1. It should provide collision free trajectories in a finite time;
2. It should be scalable with the number of spacecraft;
3. The calculations should be performed in a distributed way.

This question will be discussed with help of the following sub-questions:

1. How does the fuel consumption of a collision-free reconfiguration maneuver depend on the degree of cooperation within the formation?
2. Would J-ALM be better suited to solve the collision-free trajectory optimization problem than dual decomposition and Consensus ADMM?
3. Under what conditions does an asynchronous implementation of a dual method converge?
4. How do acceleration techniques such as HBA, FISTA and AA influence the convergence rate of both the synchronous and asynchronous dual methods?
5. Would dual methods be a feasible alternative to diminished cooperation algorithms for solving the collision-free trajectory optimization problem in real time?

In the next section, the accomplishments of the research are summarized.

1-3 Main Contributions

The research discussed in this thesis has four main contributions.

The development of a hybrid cooperative trajectory optimization algorithm which is scalable, flexible, and results in a near-optimal solution This algorithm combines a distributed algorithm in which the spacecraft do not cooperate with a centralized algorithm in which all spacecraft fully cooperate. In the resulting hybrid algorithm spacecraft on track to collide from pairs which solve a small centralized problem together, all other spacecraft solve the distributed algorithm without cooperation. This algorithm approaches the solution of the centralized algorithm, while the problem complexity approaches the distributed algorithm.

The development of an algorithm based on J-ALM for collision-free trajectory optimization This algorithm converges at a higher rate than two dual methods which have been applied to the collision-free trajectory optimization problem before, which are dual decomposition and Consensus ADMM. The calculations in J-ALM can distributed and performed in parallel by all spacecraft.

The demonstration of convergence of asynchronous J-ALM J-ALM still converges in an asynchronous setting if the stepsize suitable for synchronous J-ALM is divided by the maximum delay. This can potentially lower the convergence time of the algorithms when communication is subject to delays or spacecraft require different amounts of time to solve their part of the decoupled trajectory optimization problem.

The implementation of Anderson Acceleration on dual methods Anderson Acceleration manages to significantly increase the convergence rate of all synchronous dual methods. It outperforms both HBA and FISTA. This is the first time that Anderson Acceleration is used to accelerate an Augmented Lagrangian Method.

1-4 Thesis Outline

In Chapter 2 the collision-free trajectory optimization problem is defined in mathematical terms. All constraints and their convex approximations are discussed, followed by a short explanation of the convex problem with SCP. The chapter concludes with specifics about the simulations performed throughout this research. Next, Chapter 3 discusses three algorithms with varying cooperation: a centralized benchmark algorithm, a distributed algorithm which removes the cooperation within the formation and a hybrid algorithm which combines the centralized and distributed algorithms. The three algorithms are compared on their total required velocity change and computation times for different reconfigurations. Distributing the collision-free trajectory optimization problem using dual methods is the topic of Chapter 4. These methods are all able to arrive at the same solution as the centralized algorithm but in an iterative manner with varying convergence rates. The chapter ends with a discussion of an asynchronous implementation of one of the dual methods. In Chapter 5 different acceleration methods are discussed and simulated. These accelerated algorithms are then combined with SCP to be able to compare them to the hybrid cooperation algorithm. Finally, Chapter 6 concludes on the research, answers the research questions and provides recommendations for future work.

Chapter 2

Problem Formulation

This chapter discusses the mathematical description of the trajectory optimization problem. The non-convex problem is given in Section 2-1. In Sections 2-2 and 2-3 the convex approximations of the dynamic and collision avoidance constraint are discussed respectively. In Section 2-4 the resulting convex problem is given, together with an explanation of the SCP algorithm. In Section 2-5 an overview is given of simulation scenarios and parameters, which will be used in the simulations throughout this thesis. In Section 2-6 the most important parts of this chapter are summarized and concluded on.

2-1 The Non-convex Trajectory Optimization Problem

The non-convex trajectory optimization problem is given in (2-1). Vector $u_{i,t} \in \mathbb{R}^3$ is the 3 DOF control acceleration vector of spacecraft i at time t. The total number of spacecraft is denoted with N, and T_f is the time at which all spacecraft should have acquired their desired final states $x_{i,\text{fin}}$. As this research assumes homogeneous picosatellites for which fuel is limited, the objective is to minimize the total required accelerations. This can easily be related to the total required force (and thus to fuel consumption) by multiplying the total acceleration with the mass of a spacecraft. The first constraint enforces the initial and final states; here vector $x_{i,t} \in \mathbb{R}^6$ is the state vector of spacecraft i at time t consisting of positions and velocities in three directions, and vector $x_{i,\text{in}}$ is the given initial state. Both the initial and the final states are defined with respect to the formation center. The second constraint enforces the dynamics, the third constraint bounds the maximum acceleration per DOF by $u_{\max}$ and the final constraint is the collision avoidance constraint, with matrix $G = [\mathrm{I}_{3\times3}, 0_{3\times3}]$ where $0_{3\times3}$ is a zero matrix with size 3 by 3 and $\mathrm{I}_{3\times3}$ is the identity matrix. Matrix G thus selects only the positions from the state vectors. Finally R_{col} denotes the minimum separation distance between two spacecraft to prevent collisions.

$$\begin{aligned} \min_{u_i \ \forall i} \quad & \frac{1}{2}\sum_{i=1}^{N}\int_{t=0}^{T_f} u_{i,t}^T \ u_{i,t} \\ \text{subject to} \quad & x_{i,T_f} = x_{i,\text{fin}}, \ x_{i,0} = x_{i,\text{in}}, \ \forall i \\ & \dot{x}_{i,t} = f(x_{i,t}, u_{i,t}), \ t \in [0, T_f), \ \forall i \\ & \|u_{i,t}\|_\infty \leq u_{\max}, \ t \in [0, T_f), \ \forall i \\ & \|G(x_{j,t} - x_{i,t})\|_2 \geq R_{\text{col}}, \ t \in [0, T_f), \ \forall \ i, j \end{aligned} \tag{2-1}$$

As a solution to this problem has to be found by the spacecraft in real time, this non-convex problem is transformed to a convex problem. Transforming the constraints to convex ones will be topic of the next sections, starting with the dynamical constraint.

2-2 Initial, Final and Dynamical Constraint

To find a control acceleration vector which brings a spacecraft from its initial to the desired final state, a relative dynamical model is required. First the relevant reference frames and environmental perturbations in LEO are introduced. Then a short overview is given of possible choices for the dynamical model, followed by a detailed explanation of the chosen one. The discussion is continued with the linearization and discretization of this model. Finally, this section discusses the concept of Passive Periodic Relative Orbits (PPRO), which will be used to generate initial and final states for the spacecraft.

2-2-1 Reference Frames

To describe the relative dynamics of a spacecraft with respect to another spacecraft or a formation center, two reference frames are required.

The first reference frame, which describes the dynamics of the formation center with respect to the center of mass of the Earth, is the Earth Centered Initial (ECI) frame. As this frame is inertial, it does not rotate or accelerate. The $\hat{X}$ vector is directed to the vernal equinox, the $\hat{Z}$ vector is directed at the geographic North Pole and the $\hat{Y}$ axis completes the right-handed coordinate system [29].

The second required reference frame is the LVLH frame, which describes the states of a spacecraft in the formation with respect to the formation center. The origin of this frame lies in the center of the formation. The $\hat{x}$ vector points in the radial direction which is always directed away from the Earth, the $\hat{z}$ vector points in the cross-track direction and it is aligned with the angular momentum vector, and the $\hat{y}$ vector points in the along-track direction and completes the right handed coordinate frame. Both frames are visualized in Figure 2-1. The remaining symbols in the figure are required to describe the dynamics of both the formation center and the spacecraft; they are discussed in Section 2-2-4.

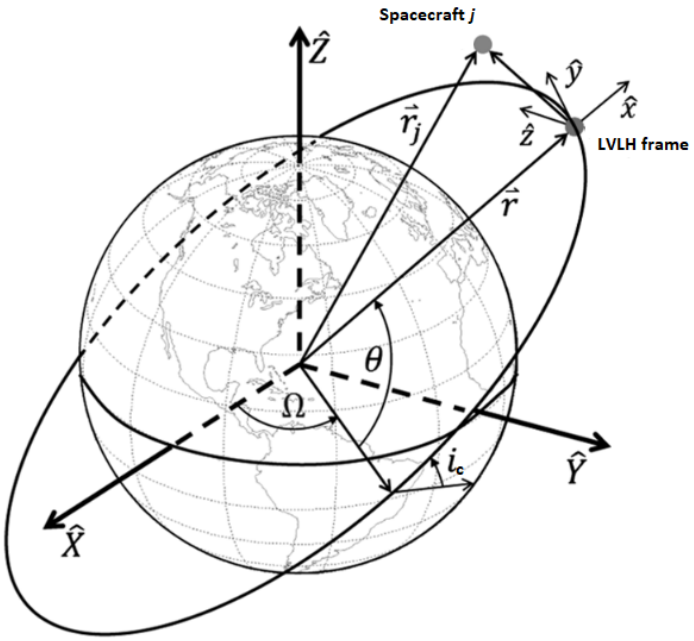


Figure 2-1: Visualization of the ECI frame $(\hat{X}, \hat{Y}, \hat{Z})$ and the LVLH frame $(\hat{x}, \hat{y}, \hat{z})$, adapted and adjusted from [2].

2-2-2 Perturbations in Low Earth Orbit

The motion of a spacecraft around Earth is perturbed mainly by three factors: a variance in the gravity potential, atmospheric drag and solar pressure. The variance in gravity potential is caused by an unevenly mass distribution of Earth [30]. It can be modeled by a higher order potential function. The dominant part of this function is the second order effect, also called the J2 perturbation. This is caused by the fact that the Earth is not a perfect sphere: it has a bulge around the equator and is thus described as an oblate spheroid. This J2 potential has been modeled accurately [30] and can be included in dynamical models for spacecraft trajectories. It is shown that the J2 perturbation has significant influence if the orbit altitude is less than 800 km.

The second perturbation of a spacecraft orbit is atmospheric drag. This effect decreases with altitude, but is shown to be still one of the dominant perturbations up to an altitude of 800 km [31]. The exact drag force depends on the atmospheric density at the specific altitude, the velocity of a spacecraft, the drag coefficient, and the surface area of the spacecraft in contact with Earth's atmosphere. This means that the drag varies with attitude, which makes it difficult to model accurately; especially since the attitude is perturbed by itself due to various disturbance torques such as the residual magnetic dipole torque, the atmospheric torque and the gravity-gradient torque [31].

The third orbit perturbation is the solar radiation force: when the light of the sun falls on the surface of a spacecraft it loses energy and therefore exerts a force on the spacecraft [31]. The resulting acceleration is a function of the surface area of the spacecraft in contact with sunlight, the reflectivity of the spacecraft material, the distance of the spacecraft to the sun

and the angle between the sunlight and the spacecraft surface. Clearly this perturbation disappears when a spacecraft is flying in the shade of another planetary body. Generally speaking, for altitudes above 800 km this perturbation becomes dominant [31].

At an altitude of 500 km, the J2 and atmospheric drag perturbations are thus dominant. Simulations show that for a formation with hundreds of 100 gram spacecraft at this altitude, neglecting both the J2 potential and the atmospheric drag results in an offset of approximately 15 m per orbit per spacecraft when compared to a model which does include both perturbations. Neglecting only the drag results in an offset of approximately 1.5 m per orbit [32].

To accurately compare trajectory optimization algorithms on their ability to minimize fuel consumption it is important to take the dominant perturbations into account. As the reconfigurations will be performed on the scale hundreds of meters and within one orbit, it is expected that J2 is a dominant perturbation, but that including atmospheric drag and solar pressure does not have a significant effect on the simulations. Therefore, the relative dynamics model used in this research will only include the J2 potential.

In case a highly accurate controller is to be developed for real-time trajectory optimization, which is one of the other open challenges for formation flying with picosatellites, at least the atmospheric drag model should definitely be included, as an offset of 1.5 meters per orbit is significant when meter or centimeter control accuracy is required. Specific simulations should then also be performed to investigate the exact effect of the solar-pressure perturbation. Designing such high accuracy controller is however outside the scope of this research.

2-2-3 Models in Literature

Over the years, many different relative dynamics models have been developed, ranging in complexity and accuracy. An extensive survey and assessment of models is given in [33] and [34]. In the first survey, it is shown that models based on Relative Orbital Elements (ROE) can describe the relative dynamics accurately. Examples are the State-Transition Matrix (STM) developed by by Gaias and D'Amico [35] and the STM by Koenig and D'Amico [36]. However, these models describe the relative motion of spacecraft in the ECI frame instead of the LVLH frame, which requires extra transformations in the trajectory optimization algorithms. In the survey by Wang [34], it is shown that the most accurate model describing relative dynamics directly in the LVLH frame is the nonlinear exact Xu-Wang model [37]. It takes the J2 perturbation into account, but neglects atmospheric drag and solar pressure. This model is relatively simple: it can be easily linearized to obtain a system with 11 first order equations.

2-2-4 The Xu-Wang Relative Dynamics Model

The Xu-Wang relative dynamics model consists of two parts. The first five equations describe the dynamics of the center of the formation, i.e. the origin of the LVLH frame, in the ECI frame. These dynamics are described using compact Reference Satellite Variables (RSV), a new set of variables introduced by Xu and Wang. The last 6 equations establish the spacecraft dynamic equations in the LVLH frame, which are dependent on the RSV and thus parameter varying. As this model is simple yet accurate, it will be used in the trajectory optimization algorithms throughout this thesis and explained in detail in the next sections.

Reference Satellite Variables

In 2008, Xu and Wang proposed a new set of variables to describe the motion of a spacecraft orbiting a planetary body, which they called the Reference Satellite Variables (RSV) [37]. This set of variables is not the most simple method to describe orbit dynamics, but has the advantage that the relative dynamics of a second spacecraft relative to the first spacecraft can be described as a function of only the RSV.

The original model uses six differential equations to describe the formation center dynamics in RSV in the rotating LVLH frame. These equations are shown in (2-2), where $\bar{r}$ is the position vector of the formation center with respect to Earth, $\hat{r}$ is the magnitude of this position vector, v_x is the radial velocity, h is the angular momentum vector, i_c is the inclination, θ is the true anomaly and Ω is the Right Ascension of Ascending Node (RAAN). Furthermore, μ is the gravitational parameter of the Earth and $c_{J2} = 3J_2\mu R_e^2/2$ is defined for notation purposes, where J_2 is the second zonal harmonic coefficient of the Earth, and R_e is the Earth's equatorial radius. Furthermore, the trigonometric functions are denoted as $\sin_\theta \equiv \sin(\theta)$. As the first five differential equations are independent of Ω, the reference satellite can be entirely described by the set $\{\bar{r}, v_x, h, \theta, i_c\}$, which forms the first 5 equations of the Xu-Wang model and are also called the Compact RSV.

$$\begin{aligned}
\dot{\hat{r}} &= v_{\hat{x}} \\
\dot{v}_{\hat{x}} &= -\frac{\mu}{\hat{r}^2} + \frac{h^2}{\hat{r}^3} - \frac{c_{J2}}{\hat{r}^4}(1 - 3\sin_{i_c}^2 \sin_\theta^2) \\
\dot{h} &= -\frac{c_{J2}\sin_{i_c}^2 \sin_{2\theta}^2}{\hat{r}^3} \\
\dot{\theta} &= \frac{h}{\hat{r}^2} + \frac{2c_{J2}\cos_{i_c}^2 \sin_\theta^2}{h\hat{r}^3} \\
\dot{i}_c &= -\frac{c_{J2}\sin_{2i_c} \sin_{2\theta}}{2h\hat{r}^3} \\
\dot{\Omega} &= -\frac{2c_{J2}\cos_{i_c} \sin_\theta^2}{h\hat{r}^3}
\end{aligned} \tag{2-2}$$

Relative Dynamics

The relative dynamics for spacecraft j with respect to the formation center are based on the Lagrange equation given in (2-3), where the Lagrangian is defined as $L = K - P$, i.e. the difference between kinetic and potential energy.

$$\frac{d}{dt}\left(\frac{\partial L_j}{\partial \dot{\bar{r}}_j}\right) - \frac{\partial L_j}{\partial \bar{r}_j} = u_j \tag{2-3}$$

The kinetic energy and potential energy of the jth spacecraft can be calculated as is shown in (2-4). Here $\bar{r}_j$ is the position vector of spacecraft j in the ECI frame and γ is the Geocentric latitude.

$$\begin{aligned} K_j &= \frac{1}{2}\dot{\hat{r}}_j^T \dot{\hat{r}}_j \\ P_{\text{grav,J2}} &= -\frac{\mu}{\hat{r}} - \frac{c_{J2}}{\hat{r}^3}\left(\frac{1}{3} - \sin_\gamma^2\right) \end{aligned} \tag{2-4}$$

By substituting (2-4) in (2-3), a nonlinear exact relative dynamics model can be obtained. This nonlinear model is given and explained in detail in [37]. For the purpose of designing a trajectory optimization algorithm, the model has to be linearized and discretized. In the next section, this linearized model will be given in detail.

2-2-5 Linearization and Discretization

The linearized relative dynamics equations are given in (2-5). The model consists of six first-order equations, describing position and velocity of a spacecraft with respect to the formation center, i.e. the center of the LVLH frame [34]. The parameters of this model are the compact RSV, so the model is linear parameter-varying.

$$\dot{x}_j = Ax_j + Bu_j \tag{2-5}$$

with

$$\begin{aligned} A &= \begin{bmatrix} 0_{3\times3} & \mathrm{I}_{3\times3} \\ A_1 & A_2 \end{bmatrix}, \quad B = \begin{bmatrix} 0_{3\times3} \\ \mathrm{I}_{3\times3} \end{bmatrix} \\ A_1 &= \begin{bmatrix} 2\frac{\mu}{r^3} + \frac{h^2}{r^4} + \frac{4c_{J2}(1-3\sin_{i_c}^2\sin_\theta^2)}{r^5} & \frac{-2v_x h}{r^3} + \frac{3c_{J2}\sin_{i_c}^2\sin_{2\theta}}{r^5} & \frac{5c_{J2}\sin_{2i_c}\sin_\theta}{r^5} \\ \frac{2v_x h}{r^3} + \frac{5c_{J2}\sin_{i_c}^2\sin_{2\theta}}{r^5} & -\frac{\mu}{r^3} + \frac{h^2}{r^4} - \frac{c_{J2}(1+2\sin_{i_c}^2-7\sin_{i_c}^2\sin_\theta^2)}{r^5} & \frac{3c_{J2}v_x\sin_{2i_c}\sin_\theta}{r^4 h} - \frac{2c_{J2}\sin_{2i_c}\cos_\theta}{r^5} \\ \frac{5c_{J2}\sin_{2i_c}\sin_\theta}{r^5} & -\frac{3c_{J2}v_x\sin_{2i_c}\sin_\theta}{r^4 h} & -\frac{\mu}{r^3} - \frac{c_{J2}(3-2\sin_{i_c}^2-5\sin_{i_c}^2\sin_\theta^2)}{r^5} \end{bmatrix} \\ A_2 &= \begin{bmatrix} 0 & \frac{2h}{r^2} & 0 \\ -\frac{2h}{r^2} & 0 & -\frac{2c_{J2}\sin_{2i_c}\sin_\theta}{r^3 h} \\ 0 & \frac{2c_{J2}\sin_{2i_c}\sin_\theta}{r^3 h} & 0 \end{bmatrix} \end{aligned} \tag{2-6}$$

This linear relative dynamics system can be discretized using a zero-order hold assumption:

$$\begin{aligned} u_j(t) = u_{j,\kappa}, \quad t \in [t_\kappa, t_{\kappa+1}), \quad \kappa = 0, \ldots, T_f - 1 \\ \text{with} \quad \Delta t = t_{\kappa+1} - t_\kappa, \quad t_f = T_f \Delta t \end{aligned} \tag{2-7}$$

Here κ is the discrete time instance. This results in

$$\begin{aligned} x_{j,\kappa+1} &= A_\kappa x_{j,\kappa} + B_\kappa u_{j,\kappa} \quad \kappa = 0, \ldots, T_f - 1 \\ \text{with} \quad A_\kappa &= e^{A(t_\kappa)\Delta t}, \quad B_\kappa = \int_0^{\Delta t} e^{A(t_\kappa)\tau} B d\tau \end{aligned} \tag{2-8}$$

A convenient way to quickly calculate the entire trajectory is to stack the iterations of (2-8) in large matrices as follows, where $X_j \in \mathbb{R}^{6(T_f-1)}$ is a vector containing all positions and velocities of spacecraft j for $\kappa = 1, \ldots, T_f$ and vector $U_j = [u_{j,0}^T, \ldots, u_{j,T_f-1}^T]^T \in \mathbb{R}^{3(T_f-1)}$ is the total control acceleration vector from time $\kappa = 0, \ldots, T_f - 1$.

$$X_j = Ox_{j,0} + TU_j \tag{2-9}$$

Matrices O and T are defined as follows:

$$O = \begin{bmatrix} A_1 \\ A_2A_1 \\ \vdots \\ \prod_{\kappa=T_f-1}^{1} A_k \end{bmatrix}, \quad T = \begin{bmatrix} B_1 & 0 & \dots & 0 \\ A_2B_1 & B_2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ \prod_{\kappa=T_f-2}^{1} A_kB_1 & \prod_{\kappa=T_f-3}^{1} A_kB_2 & \dots & B_{T_f-1} \end{bmatrix} \tag{2-10}$$

The final state can then easily be obtained by only using the last block row of O and T: $\bar{O}$ and $\bar{T}$:

$$\begin{aligned} x_{j,f} &= \bar{O}x_{j,0} + \bar{T}U_j \\ &= \begin{bmatrix} \prod_{\kappa=T_f-1}^{1} A_k \end{bmatrix} x_{j,0} + \begin{bmatrix} \prod_{\kappa=T_f-2}^{1} A_kB_1 & \prod_{\kappa=T_f-3}^{1} A_kB_2 & \dots & B_{T_f-1} \end{bmatrix} U_j \end{aligned} \tag{2-11}$$

With this dynamical constraint, the initial and final state can be linked using the control acceleration vector. To ensure that no collision occur also before and after the reconfiguration, the initial and final states are chosen to be on PPRO, which are shortly explained next.

2-2-6 Passive Periodic Relative Orbits

Fuel-efficient spacecraft formation keeping trajectories can be designed using PPRO. Passive relative orbits are the relative trajectories spacecraft follow when no actuation is applied. When these are periodic, this means that the formation remains bounded without any control effort. In [32], Morgan et al. showed that with a specific set of initial conditions the average drift of all spacecraft in a formation was only 7.55 mm/orbit. This was obtained by enforcing all spacecraft to rotate around the same point, i.e. the center of the formation, and by energy matching all spacecraft using the exact J2 perturbed Xu-Wang model. This set of conditions to obtain PPRO given in [32] will be used to generate the initial and final states for the reconfiguration maneuver, to make sure that both before and after the reconfiguration maneuver the spacecraft do not collide.

2-3 A Convex Collision Avoidance Constraint

The collision avoidance constraint as defined in (2-1) is non-convex, as it forms a sphere around each spacecraft which other spacecraft may not enter. As such nonlinear constraint results in many possible solutions and large computational complexity of the problem, the non-convex problem cannot be solved in real time for larger formations. Therefore the collision avoidance constraint has to be approximated by a convex one.

If all spacecraft have full responsibility to avoid collisions, the obtained fuel cost will be lowest as the spacecraft use maximum cooperation: some spacecraft can increase their own

fuel consumption in order to decrease the total consumption. For such problem, the constraint is approximated as follows, where $\bar{x}$ denotes the best available predicted state:

$$(\bar{x}_{j,\kappa} - \bar{x}_{i,\kappa})^T G^T G(x_{j,\kappa} - x_{i,\kappa}) \geq R_{\text{col}} \left\| G(\bar{x}_{j,\kappa} - \bar{x}_{i,\kappa}) \right\|_2 \quad \forall\ \kappa = 0, \ldots, T_f - 1 \tag{2-12}$$

Proof that this approximation is sufficient to guarantee collision free trajectories is given in [2] or [17]. In standard notation, the constraint can be expressed as a function of the control vectors:

$$\begin{aligned} A_{ij}(U_j - U_i) - B_{ij} &\leq 0 \\ \text{with } A_{ij} &= M_{ij}\hat{G}T \\ B_{ij} &= M_{ij}\hat{G}O(x_{j,0} - x_{i,0}) - R_{\text{col}}D_{ij} \end{aligned} \tag{2-13}$$

Here, matrix $M_{ij} \in \mathbb{R}^{(T_f-1)\times 3(T_f-1)}$ is a block diagonal matrix consisting of the position differences of the predicted trajectories $G(\bar{x}_{j,\kappa} - \bar{x}_{i,\kappa})$. Matrix $\hat{G}$ is a block diagonal matrix consisting of copies of G, and vector $D_{ij} \in \mathbb{R}^{T_f-1}$ contains the 2-norm of the position difference of the predicted trajectories $\|G(\bar{x}_{j,\kappa} - \bar{x}_{i,\kappa})\|_2$. Note that $A_{ij}(U_j - U_i) - B_{ij}$ is equivalent to $A_{ji}(U_i - U_j) - B_{ji}$. This collision avoidance constraint can be interpreted as two parallel planes separated by distance R_{col}, oriented perpendicular to the line connecting the predicted positions of the two spacecraft. For each sampling instance $\kappa = 0, \ldots, T_f - 1$, a new pair of planes is introduced.

Introducing these planes instead of spheres decreases the set of possible trajectories in a reconfiguration. Therefore, using the convex constraint might result in a higher total fuel consumption than when using the original constraint. However, when the entire optimization problem is convex, there is only one optimal solution which results in deterministic solutions.

2-4 The Convex Problem with Sequential Convex Programming

By discretizing the control acceleration vector as a piece-wise continuous vector and by using the dynamic constraint given in (2-11) and the collision avoidance constraint given in (2-12), the following convex problem is obtained:

$$\begin{aligned} \min_{U_i\ \forall i} \quad & \frac{1}{2}\sum_{i=1}^{N} U_i^T U_i \\ \text{subject to} \quad & x_{i,f} = \bar{O}x_{i,0} + \bar{T}U_i,\ \forall i \\ & \|U_i\|_\infty \leq u_{\max},\ \forall i \\ & A_{ij}(U_j - U_i) - B_{ij} \leq 0,\ \forall\ i, j \end{aligned} \tag{2-14}$$

As the dynamical constraints and the maximum control acceleration constraints are defined for each spacecraft separately they will be abbreviated with the following constraint: $U_i \in \mathcal{U}_i$, where $\mathcal{U}_i$ is the convex set of vectors U_i which satisfy both constraints.

In (2-14) the collision avoidance constraint is approximated using the best available solution of the trajectories. To make sure that this solution is accurate, SCP is used: an iterative method which approximates the collision avoidance constraint using the solution of the last

iteration. The algorithm is warm-started using the solution of (2-14) without the collision avoidance constraint, which results in a decoupled problem. SCP continues to iterate until the maximum change in a trajectory at all times κ is smaller than a specified tolerance ϵ_{SCP}:

$$\left\| G(x^l_{j,\kappa} - x^{l-1}_{j,\kappa}) \right\|_2 < \epsilon_{\text{SCP}},\ \kappa = 0, \dots, T_f - 1 \tag{2-15}$$

Here l indicates the SCP iteration number. To distinguish it from the discrete time indice κ in the subscript, l will be added as a superscript.

Characteristics of the problem

Let $\mathbb{R}^d = (-u_{\max}, u_{\max})$ denote the domain of $f(u)$. Objective function $f : \mathbb{R}^d \to \mathbb{R}$ as given in (2-14) is convex and differentiable, and its gradient is denoted with $\nabla f(u)$. We say that a function is strongly convex with parameter $m > 0$ if for all $u_1, u_2 \in \mathbb{R}^d$:

$$f(u_1) \geq f(u_2) + \nabla f(u_2)^T (u_1 - u_2) + \frac{m}{2} \|u_1 - u_2\|^2 \tag{2-16}$$

As $f(u)$ is a quadratic function of the form $\frac{1}{2} u^T Q u$, where Q in this case is the identity matrix, it is strongly convex with parameter $m = 1$. This can be interpreted as a quadratic lower bound to the objective function. If instead of the quadratic objective function an objective with the 1-norm of the control acceleration vector is chosen, this objective is not strongly convex anymore but only convex. Minimizing the quadratic function will result in a control acceleration vector with many nonzero elements, but each element is relatively small as the element squared contributes to the cost. Minimizing the objective with the 1-norm of the accelerations will result in a sparse control acceleration vector, but the nonzero terms will be relatively large. This will be discussed in more detail in Section 3-4. The choice depends on the specific spacecraft architecture [2], but as the quadratic function has stronger convergence properties this will be used throughout this thesis.
A function is Lipschitz continuous with constant D on a convex domain iff:

$$f(u_1) \leq f(u_2) + \nabla f(u_2)^T (u_1 - u_2) + \frac{D}{2} \|u_1 - u_2\|^2 \tag{2-17}$$

In the case of (2-14), $f(u)$ is Lipschitz continuous with $D = u_{\max}$ on the domain $\mathbb{R}^d$. This can be interpreted as a quadratic upper bound to the objective function in the specific domain.

The convergence of algorithms in this research will mainly be shown through simulations. Most of the simulation parameters will remain constant for all simulations performed in this research: they are given in the next section.

2-5 Simulation Specifics

The simulations shown and discussed in this thesis are performed to compare the convergence characteristics of different algorithms. In Chapter 3 of main interest are the total required velocity change of the collision-free trajectories and the computation times of the solver. In Chapter 4 the dual methods are compared on convergence rate, the shape of the convergence (monotonic or oscillating convergence or divergence) and the computation times of solvers. As the convergence characteristics are dependent on the specific reconfiguration maneuver, which are randomly generated in this research, Monte Carlo simulations are required to make quantitative comparisons. However, this research does not aim to provide a ready-made algorithm for a specific mission but rather gives a proof of concept of various algorithms and provides insight in the parameter selection process. Therefore it is decided to define one reconfiguration scenario, i.e. one specific set of initial and final states, and simulate all algorithms on this scenario. The simulations can thus serve as a starting point for the design of a mission-specific trajectory optimization algorithm, or can provide insight when selecting a dual method to solve a new problem.

First, the orbit initialization and simulation parameter values are discussed. Then the solver choice and implementation of the problem is shortly discussed.

Parameter values

The formation center is initialized at an altitude of 500 km. One orbit takes 5677 seconds at this altitude. The reconfiguration maneuver should be performed in $t_f = 5520$ seconds. This is selected to have exactly 23 sampling instances in one reconfiguration maneuver with a sampling time of $\Delta t = 240s$, which means that at sampling instance 23 the final configuration should be obtained. The Reference Satellite Variables are initialized by $\{r, v_x, h, \theta, i_c, \Omega\} = \{6878 \text{ km},\ 0,\ 2\pi/t_f \text{ km}^2/s,\ 0,\ 45 \text{ deg},\ 60 \text{ deg}\}$.

The positions of the initial and final states are drawn from a random distribution with standard deviation σ around the formation center. In the few cases when multiple scenarios are required, different sets of initial and final states are obtained by shuffling the random number generator of MATLAB and saving the resulting seed. For a formation of four spacecraft $\sigma = 0.1$ km, and for a formation of 10 spacecraft $\sigma = 0.3$ km. In the initial and final configurations, the positions of the spacecraft are selected to be at least distance R_{col} apart.

The collision free radius R_{col} is selected to be 0.15 km. It is chosen this large as it is only enforced at the discrete time instances. The communication distance R_{comm} is chosen to be 100 km such that all spacecraft can communicate with each other. The maximum thruster acceleration in one DOF is 1×10^{-4} km/s^2. The convergence tolerance for SCP is chosen to be $\epsilon_{\text{SCP}} = 1 \times 10^{-3}$, which corresponds to a maximum change in trajectory between SCP iterations of 1 meter.

To compare the fuel consumption of different algorithms and reconfiguration maneuvers, the average required velocity change to obtain the reconfiguration is introduced, i.e. $\Delta\bar{V} = \frac{\sum_{i=1}^{N} \|U_i\|_1 \Delta t}{N}$. The total velocity change to perform the reconfiguration can then easily be calculated as $\Delta V = \Delta\bar{V}N$. The required force to perform a reconfiguration can be obtained by multiplying the control accelerations with the mass of a spacecraft.

Solver specifics

In this research, all numerical simulations are performed using the Gurobi solver [38] in MATLAB. This solver is able to solve linear, quadratic, and mixed integer models using two main algorithms: a simplex algorithm and a barrier algorithm [39]. The choice for specific algorithm depends on the problem definition and the numerical sensitivity of the model.

In general, according to the Gurobi documentation the barrier algorithm is the fastest option, but the dual simplex algorithm is least sensitive to numerical errors [39]. In (2-14), the control acceleration vector is given in units km/s^2. This results in an optimal solution with $\frac{1}{2}u_i^T u_i$ approximately of order $O(10^{-14})$. To decrease the risk of numerical errors when such high accuracy is required, the dual simplex algorithm is selected. Also, the units of the control acceleration vector are changed to m/s^2 when used in the solver, to decrease the objective coefficient range. The computation times of both algorithms were comparable for simulations with $\Delta t = 240s$ and $N = 4$, indicating that such problem does not yet qualify as a large scale problem.

The computer used for the simulations uses a 2.5 GHz Intel Core i5 chip with 8 GB of RAM, and it runs on Windows 10. For parallel algorithms, the MATLAB "parfor" command is used, which distributes the calculations over 2 workers.

2-6 Summary and Conclusions

In this chapter first the problem description is translated to a mathematical formulation. As the problem should be solved in a distributed way by the spacecraft themselves, this non-convex problem is transformed to a convex problem by linearizing the dynamical model, discretizing the entire problem and approximating the non-convex collision avoidance constraint. The Xu-Wang dynamical model is chosen as it is an exact model which takes the J2 perturbation into account. The collision avoidance constraint is transformed to a convex form by approximating the spheres as affine planes separating the spacecraft. For this approximation, an estimation of the trajectories is required. Therefore, the convex problem is solved in an SCP framework in which the solution of each iteration is used as an estimation for the next iteration. SCP is warm-started using the solution to the convex optimization problem without the collision avoidance constraint.

Now that the convex problem is mathematically defined, different algorithms can be developed to solve the problem.

Chapter 3

Distributed Optimization with Decreased Cooperation

This chapter aims to answer the question how the fuel consumption of a collision-free reconfiguration maneuver depends on the degree of cooperation between spacecraft. First, the convex problem in combination with SCP as discussed in Section 2-4 will be solved using a centralized structure, assuming that a central unit has complete knowledge of all initial and final states of the formation. This will be the topic of Section 3-1. Then the problem will be solved in a distributed way by removing the cooperation between spacecraft. This can be done by transforming the convex collision avoidance constraint, as will be shown in Section 3-2. Third, in Section 3-3 a hybrid architecture is investigated which combines characteristics of the centralized and distributed algorithms. Finally, Section 3-4 shows simulations to compare the performance of the three algorithms.

3-1 Centralized algorithm

As in the centralized algorithm all spacecraft are responsible to avoid collisions and minimize the total cost of the entire formation, the centralized algorithm yields the optimal solution to the convex trajectory optimization problem. This centralized algorithm is given in Algorithm 1. Here $\hat{U}_i^l$ is the control acceleration vector of spacecraft i in SCP iteration l. Furthermore, $\mathbf{0}$ is used to denote a zero vector with appropriate size.

As discussed in the introduction, a centralized algorithm is unsuitable for larger formations, as in the centralized algorithm all calculations are performed by only one agent, on which the entire formation depends. Also, for larger formations even state-of-the-art solvers might not be able to solve the problem as the collision avoidance constraint scales quadratically with the number of spacecraft. To ensure safe operation also in off-nominal conditions and to make use of the computational capabilities of all spacecraft, a distributed algorithm is required.

Algorithm 1 Centralized SCP Algorithm

Initialize $U^0 = \mathbf{0}$, $l = 1$

Calculate initial $U_i^1,\ \forall i$

$$U_0^1, \ldots, U_N^1 \leftarrow \underset{U_0,\ldots,U_N}{\arg\min}\ \frac{1}{2}\sum_{i=1}^{N} U_i^T U_i$$
$$\text{s.t.} \quad U_i \in \mathcal{U}_i,\ \forall i$$

Update $X_i\ \forall i$

$$X_i^1 \leftarrow Ox_{i,0} + TU_i^1,\ \forall i$$

while $\left\| G(x_{i,\kappa}^l - x_{i,\kappa}^{l-1}) \right\|_2 \geq \epsilon_{\text{SCP}},\ \kappa = 0, \ldots, T_f - 1,\ \forall i$ **do**

$l \leftarrow l + 1$

Update collision avoidance constraints $A_{ij},\ B_{ij},\ \forall i, j$

Calculate $U_i^l,\ \forall i$

$$U_0^l, \ldots, U_N^l \leftarrow \underset{U_0,\ldots,U_N}{\arg\min}\ \frac{1}{2}\sum_{i=1}^{N} U_i^T U_i$$
$$\text{s.t.} \quad U_i \in \mathcal{U}_i,\ \forall i$$
$$A_{ij}(U_j - U_i) - B_{ij} \leq 0, \forall i, j$$

Update $X_i\ \forall i$

$$X_i^l \leftarrow Ox_{i,0} + TU_i^l$$

3-2 Distributed Algorithm without Cooperation

The convex problem given in (2-14) can not yet be solved in a distributed way due to the coupled cost and the coupled collision avoidance constraint. The cost can easily be decoupled by making each spacecraft minimize only its own cost. The collision avoidance constraint can be decoupled by replacing one of the optimization variables with the same best available solution which is used to turn the non-convex constraint into a convex one. This is shown in (3-1), where state x_i depends on optimization variable U_i and state x_j is now fixed to be $\bar{x}_j$. This constraint can be interpreted as plane again oriented perpendicular to the line connecting the two predicted positions, but now also tangent to the sphere with radius R_{col} around $\bar{x}_{j,k}$.

$$\begin{aligned}(\bar{x}_{j,\kappa} - \bar{x}_{i,\kappa})^T G^T G(\bar{x}_{j,\kappa} - x_{i,\kappa}) \geq R_{\text{col}} \left\| G(\bar{x}_{j,\kappa} - \bar{x}_{i,\kappa}) \right\|_2 \\ \forall\ \kappa = 0, \ldots, T_f - 1\end{aligned} \tag{3-1}$$

Above constraint can be transformed to standard form in a way similar way to (2-13), where $\bar{U}_j$ is the best available solution of control acceleration vector U_j:

$$A_{ij}(\bar{U}_j - U_i) - B_{ij} \leq 0 \tag{3-2}$$

To decide which spacecraft are responsible to avoid collisions, priority values $\pi_i \in \mathbb{R}_+$ are introduced [2]. Each spacecraft gets assigned a different priority value, either at random, based on the amount of fuel left in the tank or based on another criteria. A spacecraft then has to avoid all spacecraft which have a higher priority value, which means that there will always be one spacecraft which does not have to avoid any other spacecraft. When this is implemented in the SCP framework, a spacecraft keeps updating its trajectory until the SCP

convergence tolerance ϵ_{SCP} is met *and* the trajectories of all spacecraft which have to be avoided are converged already.

For each spacecraft i, the set of other spacecraft it has to avoid based on the priority values is given by $\mathcal{P}_i$. The set of spacecraft which are within communication distance R_{comm} of spacecraft i is given by $\mathcal{N}_i$. Spacecraft in set $\mathcal{N}_i$ will also be called the neighbors of i. The spacecraft which have to be avoided are thus in the set $\{\mathcal{N}_i \cap \mathcal{P}_i\}$

Using the decoupled cost together with the convex approximation of the collision avoidance constraint given in (3-2) and priority values results in the distributed problem given in (3-3). This problem can be solved separately by each spacecraft $i = 1, \ldots, n$. The total cost can be obtained by summing all costs of individual spacecraft.

$$\begin{aligned} \min_{U_i} \quad & \frac{1}{2} U_i^T U_i \\ \text{subject to} \quad & U_i \in \mathcal{U}_i \\ & A_{ij}(\bar{U}_j - U_i) - B_{ij} \leq 0, \quad j \in \{\mathcal{N}_i \cap \mathcal{P}_i\} \end{aligned} \tag{3-3}$$

The distributed algorithm with SCP is given in Algorithm 2 for one spacecraft i. As all calculations are performed on board the spacecraft and each spacecraft has to avoid only a subset of its neighbors, this algorithm is both scalable and flexible. However, as it lacks any form of cooperation in avoiding collisions, the total fuel consumption will be higher than the solution of the centralized algorithm.

3-2-1 Model Predictive Control

As a spacecraft can only detect other spacecraft when they are within its communication radius, collisions are avoided only between neighbors and only at the specific discrete time instances. A spacecraft pair which starts outside each others communication radius does not avoid each other and thus can still collide during the flight to the new configuration. This can be solved by implementing the distributed SCP algorithm in a Model Predictive Control (MPC) framework, as is demonstrated in [2]. Instead of executing the entire obtained control acceleration vector, only the control vector for the first time instance is executed. Each discrete time instance κ the SCP algorithm is then solved again from time κ_0 to $\kappa_0 + T_f - 1$ using updated initial states, set of neighbors and priority values, until the final configuration is obtained. The MPC framework makes sure that all potential collisions between spacecraft started outside each others communication radius will be taken into account later in the reconfiguration. Next to this, there is also a possibility that spacecraft start outside each others communication radius, but fly with such speed that they collide within one sampling instance. To prevent this, an extra constraint on the spacecraft velocity can be introduced: $\|[0_{3\times3}, \mathrm{I}_{3\times3}]x_{i,\kappa}\|_2 \leq v_{\max}, \kappa = k_0, \ldots, k_0 + T_f - 1$. Now Δt, R_{comm} and $v_{\max}$ can be chosen such that the velocities of two spacecraft can never be large enough to collide within Δt seconds if they start outside each others communication radius. The details of this proof can be found in [2].

An extra advantage of the MPC framework is that it adds robustness with respect to sensor and actuator uncertainties. For real-time implementations, a combination of SCP with MPC would thus be preferred.

Algorithm 2 Distributed SCP Algorithm for one spacecraft i

Initialize $U^0 = \mathbf{0},\ l = 1$

Calculate initial U_i^1

$$\hat{U}_i^1 \leftarrow \underset{U_0,\dots,U_N}{\arg\min}\ \frac{1}{2}U_i^T U_i$$
$$\text{s.t.}\quad U_i \in \mathcal{U}_i$$

Share π_i and U_i^1 with neighbors $j \in \{\mathcal{N}_i \cap \mathcal{P}_i\}$ and receive theirs

Update state trajectory X_i

$$X_i^1 \leftarrow Ox_{i,0} + TU_i^1$$

while $\left\|G(x_{i,\kappa}^l - x_{i,\kappa}^{l-1})\right\|_2 \geq \epsilon_{\text{SCP}},\ \kappa = 0,\dots,T_f - 1$ and $j \in \{\mathcal{N}_i \cap \mathcal{P}_i\}$ are not converged yet **do**

$l \leftarrow l + 1$

Update A_{ij}, B_{ij}, $\forall j$

Calculate U_i^l

$$U_i^l \leftarrow \underset{U_i}{\arg\min}\ \frac{1}{2}U_i^T U_i$$
$$\text{s.t.}\quad U_i \in \mathcal{U}_i$$
$$A_{ij}(U_j^{l-1} - U_i) - B_{ij} \leq 0,\ j \in \{\mathcal{N}_i \cap \mathcal{P}_i\}$$

Share U_i^l with neighbors $j \in \{\mathcal{N}_i \cap \mathcal{P}_i\}$ and receive theirs

Update X_i

$$X_i^l \leftarrow Ox_{i,0} + TU_i^l$$

Share with neighbors $j \in \{\mathcal{N}_i \cap \mathcal{P}_i\}$ that trajectory is converged

3-3 Hybrid Algorithm

The centralized algorithm has full cooperation between the spacecraft when avoiding collisions, but is not scalable and flexible. The distributed algorithm is scalable and flexible, but has no cooperation whatsoever. A hybrid algorithm is proposed to combine the properties of both algorithms.

This algorithm is mainly based on the algorithm without any cooperation, but here two spacecraft are allowed form a cooperating pair and solve a small centralized problem for the two of them. This could be implemented as follows. First, each spacecraft i calculates its trajectory without taking collision avoidance into account. After sharing with and receiving these trajectories from its neighbors in set $\mathcal{N}_i$, each spacecraft makes a list of neighbors it potentially collides with, $\mathcal{C}_i \in \mathcal{N}_i$, which is sorted based on the time instance of the potential collision. If two spacecraft find each other on top of this list, they form a cooperating pair. If the first spacecraft potentially causing a collision is already cooperating with another spacecraft, it checks with the next spacecraft causing a potential collision, and so forth. When two spacecraft form a cooperating pair, one of the two spacecraft solves a centralized problem for the two of them and shares the solution with the other. All spacecraft which do not form a pair, either because they are not on track to any potential collision or because all potential colliding spacecraft have already formed another pair, solve the distributed problem explained in the previous section. The priority values are still used to determine which remaining spacecraft in list $\mathcal{C}_i$ should be avoided and which not.

Essentially the hybrid algorithm thus consists of two algorithms: the cooperating pairs solve (3-4) for the two of them where the two spacecraft in the pair are denoted with $sc1$ and $sc2$, all other spacecraft solve the distributed problem (3-3). Such hybrid algorithm is expected to be particularly useful when it is implemented in the MPC framework, as then in each new MPC iteration new pairs can be formed. Of course such scheme could also be extended to larger cooperating groups, essentially creating an hierarchical trajectory optimization algorithm.

$$\begin{aligned} \min_{U_{sc1},U_{sc2}} \quad & \frac{1}{2}(U_{sc1}^T U_{sc1} + U_{sc2}^T U_{sc2}) \\ \text{subject to} \quad & U_i \in \mathcal{U}_i,\ i = sc1, sc2 \\ & A_{sc1,sc2}(U_{sc2} - U_{sc1}) - B_{sc1,sc2} \leq 0 \\ & A_{ij}(\bar{U}_j - U_i) - B_{ij} \leq 0,\ i = sc1, sc2,\ j \in \{\mathcal{N}_i \cap \mathcal{P}_i\} \end{aligned} \tag{3-4}$$

Now that three algorithms with varying cooperation have been introduced, their performance can be compared using simulations.

3-4 Simulations

First, a result of the centralized algorithm will be discussed in detail to give insight in characteristics of the solution. Then the three algorithms discussed in previous sections will be compared on different reconfiguration scenarios.

In Figure 3-1 the relative distances during a reconfiguration maneuver are shown without and with the collision avoidance constraints. The black dotted line indicates the minimum safe distance R_{col}. In the top plot, the constraint is mainly violated by spacecraft 1 and 2, but also a little by the pairs 1-4 and 2-4. In the lower plot it is clear that all relative distances remain on or above the minimum safe distance. Due to the final configuration constraint, an adjustment in one part of the relative distance plot influences the entire trajectory and thus three relative trajectories. This can be seen for example in the relative distance plots of pair 1-3: in the top plot they remain almost 0.2 km apart, but in the lower plot this distance is decreased to 0.15 km due to the adjustments for pair 1-2. As the trajectories often can not be independently altered, an algorithm can show very different convergence rates for different scenarios.

In Figure 3-2 the magnitudes of the control accelerations corresponding to Figure 3-1 are shown. Especially the accelerations of spacecraft 1 and 2 are increased, in order to increase their relative distance. The acceleration vectors are piece-wise continuous as they are discretized using the zero-order hold assumption. Furthermore, they are all nonzero due to the quadratic objective function in the problem definition. If instead the 1-norm is minimized, the resulting control acceleration vector is sparse, as is shown in Figure 3-3. It is interesting to note that for the problem with objective function $\sum_i^N \frac{1}{2} U_i^T U_i$. the required average velocity change is $\Delta\bar{V} = 0.5064$ m/s, whereas the problem with objective function $\sum_i^N \|U_i\|_1$ results in an average velocity of $\Delta\bar{V} = 0.3325$ m/s. This is a substantial difference, which definitely has to taken into account when designing the spacecraft architecture and controller.

Now that the solution characteristics are clear, the three algorithms can be compared on their convergence characteristics. First, ten different scenarios are simulated for a reconfiguration

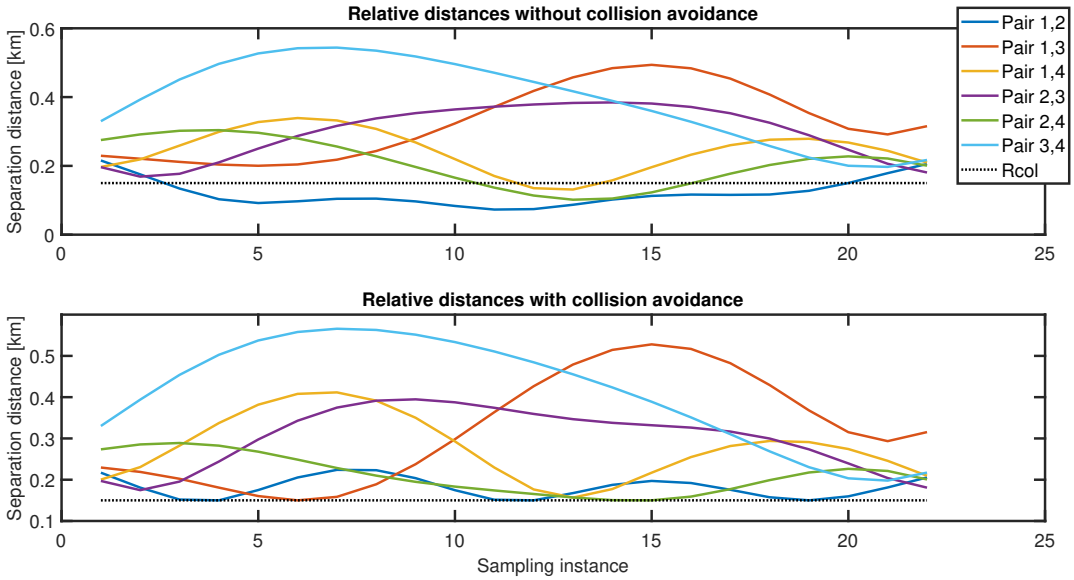


Figure 3-1: Relative distances between the spacecraft pairs before and after collisions are avoided. When the constraint is active, all relative trajectories are pushed above R_{col}.

of four spacecraft. The averaged results are listed in Table 3-1. Both the distributed and the hybrid implementation results in a larger mean $\Delta\bar{V}$ and standard deviation than the centralized implementation. The distributed algorithm also required more SCP iterations to converge on average, whereas the hybrid algorithm actually converged in fewer iterations on average. The results of the hybrid algorithm show that the combination of the distributed architecture with cooperating pairs indeed can combine centralized and distributed characteristics: the obtained solution is relatively close to the solution of the centralized architecture, whereas the calculations are distributed over almost all spacecraft. For four spacecraft, the computation times per iteration per spacecraft are similar for the three algorithms.

The distributed algorithm resulted in a higher $\Delta\bar{V}$ cost than the centralized algorithm in every one of the ten runs. This was however not the case for the hybrid algorithm: in one reconfiguration maneuver the hybrid algorithm outperformed the centralized one in terms of $\Delta\bar{V}$ cost. This shows that the centralized algorithm is not optimal in itself, which is expected as its collision avoidance constraint is still a conservative approximation which excludes a large part of the solution space. Due to the specific structure with varying responsibility to avoid collisions, the distributed and hybrid algorithm find other solutions than the centralized algorithm, which results in different convex approximations of the collision avoidance constraint in the next SCP iteration. In some cases, this might even result in a lower $\Delta\bar{V}$.

	$\Delta\bar{V}$	Standard deviation	SCP iter.	Time per iter. [s]
Centralized	0.4572 (100%)	0.1113	10.2	0.5111
Hybrid	0.4694 (103%)	0.1211	9.3	0.5506
Distributed	0.5157 (113%)	0.1460	13.2	0.4629

Table 3-1: Four spacecraft reconfiguration: Averaged convergence characteristics over 10 runs with $\sigma = 0.1$ km. The computation time is given per spacecraft per iteration.

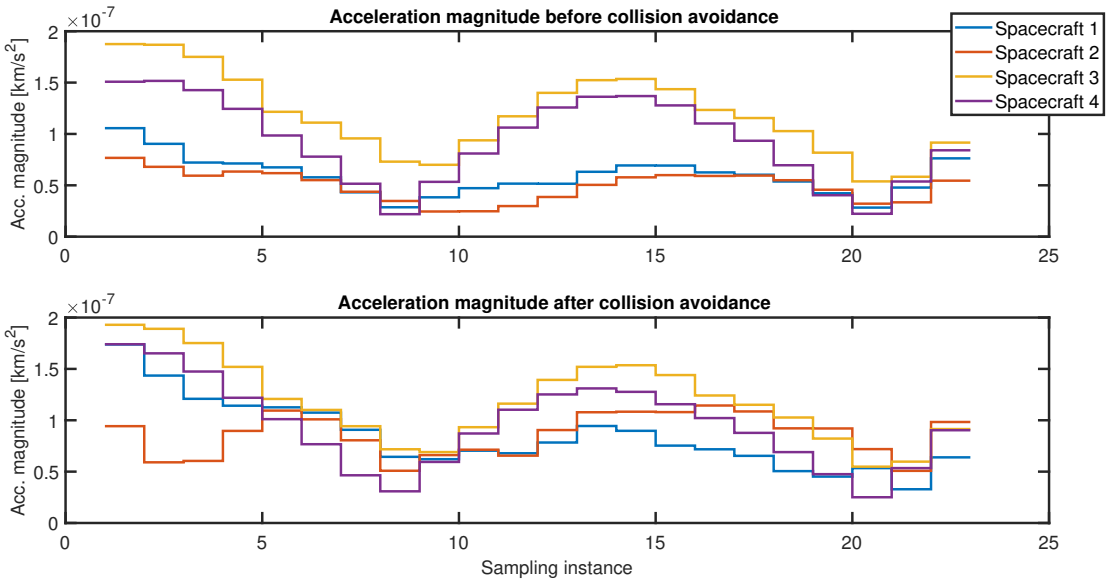


Figure 3-2: Acceleration magnitudes corresponding to Figure 3-1. As each spacecraft (Sc) has 6 DOF control, the magnitudes of the accelerations in all DOF are summed per sampling instance.

The averaged convergence characteristics for a 10 spacecraft reconfiguration are shown in Table 3-2. Again the distributed algorithm performed worse than the centralized and hybrid one, but the difference is much smaller now than with the four spacecraft reconfiguration. This can be explained by looking at the standard deviation used to randomly generate initial and final states for the reconfiguration maneuvers. The results obtained in this section depend largely on this standard deviation. If σ is chosen relatively large, the initial and final states are likely to be far away and even without the collision avoidance constraint the fuel cost will be relatively high. However, not many collisions have to be avoided during this reconfiguration. If σ is chosen relatively low, initial and final states are not that far away from each other and the spacecraft can move to their final state at a low fuel cost. Now many collisions have to be avoided however. Therefore, with relatively small σ the performance of the three algorithms will vary more than for a large σ. This means that an objective comparison of performance for general scenarios requires many different simulations. For now, we can conclude that the distributed algorithm results in solutions with a higher $\Delta\bar{V}$ than the solutions of the centralized and hybrid algorithms, but the specific performance difference depends on the chosen reconfiguration scenario. In terms of computation time, the differences between the three algorithms are more clear: in the centralized algorithm one spacecraft has to solve the entire problem whereas the hybrid and distributed algorithm share the workload with the entire formation. This results in a reduction of computation time per spacecraft per iteration.

Finally, one simulation is performed with 100 spacecraft where the initial and final states are drawn from a random distribution with $\sigma = 2$ km. If no collisions are avoided, 6 different spacecraft pairs collide during their reconfiguration trajectories. The convergence characteristics of the three algorithms are given in Table 3-3. As the centralized algorithm has to solve the entire problem again in each SCP iteration, the time to solve one iteration remains almost constant, approximately 30 seconds. The algorithm did not converge yet in iteration 71 as in each iteration some trajectories were slightly altered, even if they already converged

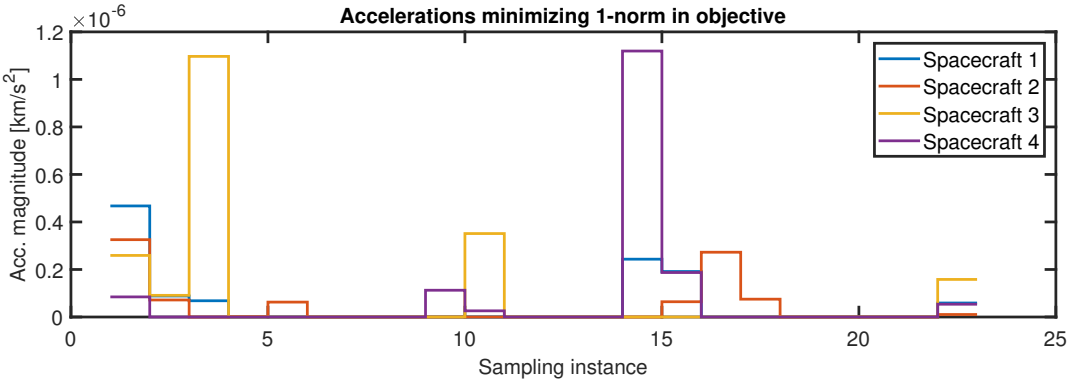


Figure 3-3: Obtained acceleration magnitudes when (2-14) is solved with objective $\sum_i^N \|U_i\|_1$ instead of $\sum_i^N \frac{1}{2} U_i^T U_i$. This results in sparse acceleration vectors.

	$\Delta\bar{V}$	Standard deviation	SCP iter.	Time per iter. [s]
Centralized	1.1260 (100%)	0.1362	12.9	0.7726
Hybrid	1.1387 (101%)	0.1413	18.1	0.4420
Distributed	1.1487 (102%)	0.1385	23.6	0.3241

Table 3-2: Ten spacecraft reconfiguration: Averaged convergence characteristics over 10 runs with $\sigma = 0.3$ km. Computation time is given per spacecraft per iteration.

in previous iterations. The hybrid and distributed algorithms did not have this problem: the converged trajectories were not allowed to change anymore. These algorithms converged in respectively 9 and 25 iterations. The computation time per spacecraft in the first iteration for the hybrid and distributed algorithm remains below one second even for this large reconfiguration. As most spacecraft do not have to avoid any collisions but still have to use much control acceleration to reach the desired final states, the relative gain of using a centralized or hybrid algorithm to avoid collisions instead of a distributed one is very small.

Another advantage of the distributed and hybrid algorithm is that it allows for fuel balancing in the formation. The priority values can be selected based on amount of fuel left per spacecraft. This way, the spacecraft with less fuel left can be spared, at the cost of a higher fuel consumption for other spacecraft. This might increase the total life-time of the formation.

	SCP iter.	Total time [s]	Time per sc. in iter. 1 [s]	$\Delta\bar{V}/\Delta\bar{V}^*$
Centralized	Stopped at 71	2085	32.1	1
Hybrid	9	143	0.5798	1.000034
Distributed	25	198	0.5691	1.000054

Table 3-3: Convergence characteristics for 100 spacecraft formation.

3-5 Summary and Conclusions

In this chapter the collision-free reconfiguration problem is solved in a distributed way by removing the cooperation between spacecraft. As in such distributed approach all spacecraft minimize only their own required velocity change, the average required velocity change over the spacecraft is higher than the one obtained using the centralized algorithm. As this difference originates from the way collisions are avoided, a hybrid algorithm is proposed in which spacecraft on track to collide form cooperating pairs. These pairs solve a small centralized problem for the two of them. Such setup is shown to reduce the total required velocity change in a reconfiguration, approaching the centralized solution. Furthermore, the hybrid algorithm is flexible and scalable with the number of spacecraft as each optimization problem remains relatively small. However, the hybrid algorithm still requires more fuel than the centralized algorithm.

There is also a possibility to obtain exactly the centralized solution, but in a distributed way using dual methods. This will be the topic of next chapter.

Chapter 4

Dual Methods

Instead of distributing the problem by removing or decreasing the cooperation in the formation, the problem can also be distributed using dual methods. This chapter answers the questions whether J-ALM would be better suited to solve the collision-free trajectory optimization problem than dual decomposition and Consensus ADMM, and how to develop an asynchronous implementation of a dual method. In dual methods not the objective function is minimized, but the Lagrangian function which consists of the objective function plus the collision avoidance constraints multiplied with dual vectors. This will be explained in detail in the following sections. First, dual decomposition is discussed, followed by a Consensus ADMM, a sequential and a parallel decomposition of the Augmented Lagrangian Method (ALM). Then, an asynchronous algorithm based on the parallel decomposition of ALM is discussed, and convergence of the algorithms is shown through simulations. All dual methods discussed in this chapter solve only the first iteration of the SCP algorithm, to be able to accurately compare the different convergence rates. The simulations in this chapter are all based on the same reconfiguration with four spacecraft, unless stated otherwise.

4-1 Dual Decomposition

In this section, first the dual decomposition algorithm is explained. Then, with the help of simulations the selection of an appropriate stepsize for the dual update is discussed. This discussion is interesting as it differs from standard subgradient theory due to the fact that the coupling is introduced in inequality instead of equality constraints.

The Lagrangian is given in (4-1). It consists of the objective function plus a collision avoidance constraint for each pair i, j multiplied by a dual vector $y_{ij} \geq 0$. This constraint comes from the fact that the collision avoidance constraints are inequality constraints. A separate dual vector is introduced for each spacecraft pair i, j. In contrast to the coupled convex trajectory optimization problem as given in (2-14) the Lagrangian in combination with the two separable constraints results in a decoupled problem: each spacecraft in the formation can minimize its

own part separately.

$$\begin{aligned} L(u,y) &= \frac{1}{2}\sum_{i=1}^{N} U_i^T U_i + \sum_{i=1}^{N-1}\sum_{j>i} y_{ij}^T[A_{ij}(U_j - U_i) - B_{ij})] \\ &= \sum_{i=1}^{N}\Big[\frac{1}{2}U_i^T U_i - \sum_{j>i} y_{ij}^T(A_{ij}U_i + B_{ij}) + \sum_{j<i} y_{ji}^T A_{ji} U_i\Big] \\ &= \sum_{i=1}^{N} L_i(u_i, y) \end{aligned} \tag{4-1}$$

In dual decomposition, minimizing the Lagrangian over u is alternated with maximizing the dual problem over y. This dual problem is defined as follows:

$$\max_{y\geq 0} \min_{u} L(u,y) \tag{4-2}$$

Maximization over y can be done using a subgradient method as residual vector $r_{ij} = A_{ij}(U_j - U_i) - B_{ij}$ is a subgradient of the Lagrangian with respect to y_{ij}. This residual vector is a measure for the collision avoidance constraint violation of spacecraft pair i, j which means that an element of y_{ij} is increased when the corresponding element of r_{ij} is positive, and decreased or kept zero otherwise. The dual vector thus acts as a price on violating the collision avoidance constraint which increases more for larger constraint violations. The dual vector is updated as follows, where k is the iteration number of the dual algorithm and ρ is the dual update stepsize:

$$y_{ij,k+1} = y_{ij,k} + \rho r_{ij,k} \tag{4-3}$$

The dual decomposition algorithm is given in Algorithm 3. Here $[.]_+$ denotes the projection operator to $\mathbb{R}_+$. We assume that there exists a finite vector U for which strict collision avoidance constraints are satisfied. With this assumption in combination with the fact that $f(u)$ is convex, Slater's condition is satisfied and there is no duality gap, which means that the solution to the dual decomposition algorithm equals the centralized solution.

Algorithm 3 Dual decomposition algorithm

Initialize $y_1 = \mathbf{0}$, $k = 0$, A_{ij}, B_{ij}, $\forall i, j$
repeat
  $k \leftarrow k + 1$
  Update U_i in parallel:
  for $i = 1, \ldots, N$ **do**
    $U_{i,k} \leftarrow \underset{U_i}{\operatorname{argmin}}\ L_i(u_i, y_k)$ s.t. $U_i \in \mathcal{U}_i$
  for all i, j **do**
    $r_{ij,k} \leftarrow A_{ij}(U_j - U_i) - B_{ij}$
    $y_{ij,k+1} \leftarrow [y_{ij,k} + \rho r_{ij,k}]_+$
until $[r_{ij,k}]_+ \in \emptyset\ \forall i, j$

As discussed, minimizing the Lagrangian over u can be solved separately by each spacecraft. However, the subgradients have to be aggregated in order to update the dual vector, a task

for which some sort of coordination is required. As each spacecraft pair has its own dual vector, these dual vectors can be updated in a distributed way as well if all spacecraft share their updated control acceleration vectors with each other. A specific communication scheme has to be introduced in order to make sure that all dual vectors are updated accurately.

As dual methods are iterative algorithms, a measure for the convergence is needed. Two residuals are introduced for this goal. The first one is the primal residual which is defined to be the maximum violation of the collision avoidance constraint, i.e. the maximum of vectors r_{ij} $\forall i, j$. The second residual is the control vector residual which will be called the dual residual. This residual measures the maximum change of an element of the control acceleration vectors. Both residuals are defined in (4-4). Note that the standard definition of the dual residual is defined for the case when $N = 2$; the dual residual then only contains the second agent (in this case U_2). This is adjusted as in this research $N \geq 2$ and all agents are equally important.

$$\begin{aligned} pr_k &= \max\ \{pr_{ij,k},\ \forall i, j\} \\ &= \max\{\max\ \{[A_{ij}(U_{j,k} - U_{i,k}) - B_{ij}]_+\},\ \forall i, j\} \\ dr_k &= \max\ \{\|U_{i,k} - U_{i,k-1}\|_\infty,\ \forall i, j\} \end{aligned} \tag{4-4}$$

For real-time implementation, these residuals can be used as stopping criteria. As the collision avoidance constraint has units $\mathrm{km}^2/\mathrm{s}^2$, a bound of $pr \leq 1 \times 10^{-6}$ corresponds to a maximum violation of the collision avoidance constraint of 1 meter. The optimal control acceleration vector typically consists of accelerations of the order $O(1\times10^{-7})$ in units km/s^2. To make sure a dual algorithm is converged, a bound can be selected of $dr \leq 1 \times 10^{-11}$, which corresponds to a changes only from the fifth nonzero element of the control acceleration vector on. In the simulations in this thesis these stopping criteria are usually not activated, as the entire convergence of the algorithms provides useful insights for the comparison of the algorithms.

As the Lagrangian and control acceleration vectors reach their solution asymptotically, they will be plotted on logarithmic scale as $|L^* - L_k|/L^*$ and $|\Delta\bar{V}^* - \Delta\bar{V}_k|/\Delta\bar{V}^*$, where $\Delta\bar{V}^*$ denotes the solution obtained with the centralized algorithm, and $\Delta\bar{V}_k$ denotes the solution of the dual algorithm at iteration number k.

4-1-1 Selecting an appropriate stepsize

In [25], the standard rules for selecting stepsize ρ are given for subgradient methods, together with a convergence proof. It is shown that a subgradient method converges as follows when $y_0 = \mathbf{0}$:

$$L_k^{\text{best}} - L^* = \frac{\|y^*\|_2^2 + \sum_{i=1}^k \rho_i^2 \|r_i\|_2^2}{2\sum_{i=1}^k \rho_i} \tag{4-5}$$

Here y^* denotes the optimal dual vector. If ρ is chosen to be square summable but not summable as shown in (4-6), the algorithm converges; i.e. $L_k^{\text{best}} \to L^*$ for $k \to \infty$.

$$\sum_{k=1}^{\infty} \rho_k = \infty, \quad \sum_{k=1}^{\infty} \rho_k^2 < \infty \tag{4-6}$$

A simulation of dual decomposition with such diminishing stepsize, specifically $\rho = \frac{5\times10^{-5}}{k}$, is shown in the blue line in Figure 4-1. On logarithmic scale, the plot remains almost constant; it is clear that it will take many more than 2000 iterations before the stopping criteria of $pr \leq 1 \times 10^{-6}$ is met.

For constant ρ, it is shown in [25] that a subgradient method converges as follows:

$$L_k^{\text{best}} - L^* = \frac{\|y^*\|_2^2 + \rho^2 \sum_{i=1}^{k} \|r_i\|_2^2}{2k\rho} \tag{4-7}$$

This shows that in case $\|r_i\|_2$ remains the same order of magnitude, say $\|r_k\|_2 \leq S,\ \forall k$ would be an appropriate approximation, then if $k \to \infty$ a residual of $\frac{S^2\rho}{2}$ remains.

As the dual variable y is initialized to be zero, the first iteration of dual decomposition does not take the collision avoidance constraint into account, and the solution is likely to be far off the optimal solution. When the algorithm is nearly converged, very detailed dual update steps have to be taken in order to reach the optimal solution with high accuracy. Therefore it makes sense to choose a stepsize which starts large but diminishes over the iterations.

However, the red line of Figure 4-1, which represents the same simulation but now with constant $\rho = 3 \times 10^{-5}$, shows that the algorithm converges at a much higher rate than the algorithm with diminishing ρ, to $pr \approx 1 \times 10^{-15}$ in approximately 3300 iterations. This shows that for this specific problem, a constant stepsize ρ is actually better suited than a diminishing stepsize.

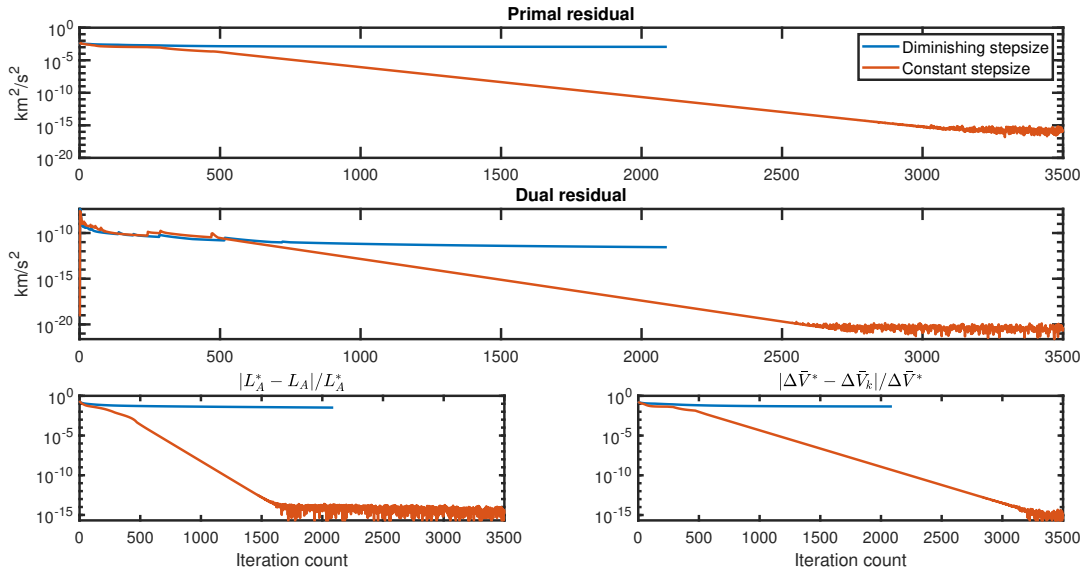


Figure 4-1: Dual decomposition with diminishing stepsize $\rho = \frac{5\times10^{-5}}{k}$ versus constant stepsize $\rho = 3 \times 10^{-5}$.

The fact that the algorithm still converges despite the constant stepsize can be explained by looking into the dual decomposition algorithm in more detail. Some authors discuss the convergence of dual decomposition with constant stepsize. For example in [40] Boyd et al. comment that a constant stepsize results in convergence if the dual function is differentiable,

but no further details are discussed. In [41], Nedic et al. develop convergence proofs and bounds for dual decomposition with constant stepsize, but their results are based on the average optimization variable (the control acceleration vector averaged over the iterations) instead of the latest optimization variable. To understand exactly what happens in dual decomposition for solving the collision free trajectory problem, this thesis discusses the topic in more detail with a focus on the value of the Lagrangian function over the iterations. Although it is likely that such discussion can be found in literature, a thorough search did not yield any similar discussion yet.

First, a shorter notation of the Lagrangian is introduced. An example of the shorter notation of the collision avoidance constraint residuals for four spacecraft is given as follows:

$$r = \begin{bmatrix} r_{12} \\ r_{13} \\ r_{14} \\ r_{23} \\ r_{24} \\ r_{34} \end{bmatrix} = \begin{bmatrix} -A_{12} & A_{12} & 0 & 0 \\ -A_{13} & 0 & A_{13} & 0 \\ -A_{14} & 0 & 0 & A_{14} \\ 0 & -A_{23} & A_{23} & 0 \\ 0 & -A_{24} & 0 & A_{24} \\ 0 & 0 & -A_{34} & A_{34} \end{bmatrix} \begin{bmatrix} U_{12} \\ U_{13} \\ U_{14} \\ U_{23} \\ U_{24} \\ U_{34} \end{bmatrix} - \begin{bmatrix} B_{12} \\ B_{13} \\ B_{14} \\ B_{23} \\ B_{24} \\ B_{34} \end{bmatrix} \tag{4-8}$$

The acceleration vectors and dual variable vectors are all stacked in one large vector as follows:

$$\begin{aligned} U &= [U_1^T, \ldots, U_i^T, \ldots, U_N^T]^T \\ y &= [y_{12}^T, y_{13}^T, \ldots, y_{N-1,N}^T]^T \end{aligned} \tag{4-9}$$

Now we can define the collision avoidance constraint residual for two spacecraft as follows:

$$r_{ij} = \Big[[F_{ij}]_1, \ldots, [F_{ij}]_N\Big]U - B_{ij} \tag{4-10}$$

with

$$\begin{aligned} [F_{ij}]_j = A_{ij},\ [F_{ij}]_i = -A_{ij}, \\ [F_{ij}]_{\{1,\ldots,N\}\setminus\{i,j\}} = 0 \end{aligned} \tag{4-11}$$

These residuals can be stacked in block rows using a separate block row for each spacecraft pair: $F = \Big[[F_{12}]^T, [F_{13}]^T, \ldots, [F_{N-1,N}]^T\Big]^T$. The B_{ij} matrices can also be stacked as follows: $B = [B_{12}^T, B_{13}^T, \ldots, B_{N-1,N}^T]^T$. Now all residual vectors can be included in the following total residual vector:

$$r = FU - B \tag{4-12}$$

This results in a shortened notation of the Lagrangian given in (4-1):

$$\begin{aligned} L(u, y) &= \frac{1}{2}U^T U + y^T(FU - B) \\ &= f(u) + y^T r \end{aligned} \tag{4-13}$$

Now the dual decomposition algorithm can be discussed in more detail. Two parameters are important for this: the function value $f(u) = \sum_i^N U_i^T U_i$ and its relation to residual vector r, and the change in value of the Lagrangian function $\Delta L_k = L_{k+1} - L_k$. As $y_0 = \mathbf{0}$, both parameters $f(u_0)$ and $L(u_0, y_0)$ start below their optimal values, and thus have to increase. Let

us look first at ΔL_k in detail. The dual decomposition algorithm consists of two optimization steps per iteration: the subgradient step which updates y to maximize the Lagrangian, and the control acceleration vector update which minimizes the Lagrangian. This is shown in (4-14). Note that the $>$ and $<$ signs are used, instead of $\geq$ and $\leq$. This is valid except for the case when the algorithm is converged, but then the stopping criteria are met anyway.

$$\begin{aligned} L(u_k, y_{k+1}) > L(u_k, y_k) \quad &\rightarrow \quad f(u_k) + y_{k+1}^T r_k > f(u_k) + y_k^T r_k \\ L(u_{k+1}, y_{k+1}) < L(u_k, y_{k+1}) \quad &\rightarrow \quad f(u_{k+1}) + y_{k+1}^T r_{k+1} < f(u_k) + y_{k+1}^T r_k \end{aligned} \tag{4-14}$$

Furthermore, as u_k minimizes $L(u, y_k)$ and $f(u)$ is strictly convex:

$$L(u_{k+1}, y_k) > L(u_k, y_k) \rightarrow \quad f(u_{k+1}) + y_k^T r_{k+1} > f(u_k) + y_k^T r_k \tag{4-15}$$

The change in value of the Lagrangian, ΔL, can thus be expressed as the maximization step plus the minimization step.

$$\Delta L_k = L(u_{k+1}, y_{k+1}) - L(u_k, y_k) = \Big(L(u_k, y_{k+1}) - L(u_k, y_k)\Big) + \Big(L(u_{k+1}, y_{k+1}) - L(u_k, y_{k+1})\Big) \tag{4-16}$$

In order to obtain an exact expression for this term, let us look at those two terms separately. The maximization step can be characterised as follows:

$$\begin{aligned} L(u_k, y_{k+1}) - L(u_k, y_k) &= f(u_k) + y_{k+1}^T r_k - (f(u_k) + y_k^T r_k) \\ &= (y_{k+1}^T - y_k^T) r_k \end{aligned} \tag{4-17}$$

For the characterization of the minimization step, the relation in (4-15) is used:

$$f(u_{k+1}) - f(u_k) > y_k^T (r_k - r_{k+1}) \tag{4-18}$$

Then, an expression for the minimization step can be obtained, using above result.

$$\begin{aligned} L(u_{k+1}, y_{k+1}) - L(u_k, y_{k+1}) &= f(u_{k+1}) + y_{k+1}^T r_{k+1} - f(u_k) - y_{k+1}^T r_k \\ &= f(u_{k+1}) - f(u_k) - y_{k+1}^T (r_k - r_{k+1}) \\ &> -(y_{k+1}^T - y_k^T)(r_k - r_{k+1}) \\ -(y_{k+1}^T - y_k^T)(r_k - r_{k+1}) < L(u_{k+1}, y_{k+1}) &- L(u_k, y_{k+1}) < 0 \end{aligned} \tag{4-19}$$

By summing (4-17) and (4-19) an expression is obtained for ΔL:

$$\Delta L_k > (y_{k+1}^T - y_k^T) r_k - (y_{k+1}^T - y_k^T)(r_k - r_{k+1}) = (y_{k+1}^T - y_k^T) r_{k+1} \tag{4-20}$$

From the first relation in (4-14) we know that $(y_{k+1}^T - y_k^T) r_k > 0$. Furthermore, as $y_{k+1} = [y_k + \rho r_k]_+$, if $\rho_k \rightarrow 0$, $y_{k+1} \rightarrow y_k$ and resulting from this, $r_{k+1} \rightarrow r_k$. Concluding, ρ can always be chosen small enough to make sure that $\Delta L_k > 0$ (except when the algorithm is converged).

What is important to notice is that the y-update increases the Lagrangian function value only by increasing the $y^T r$ term in $L = f(u) + y^T r$. Over the iterations, the minimization step thus

shifts the weight from minimizing $f(u)$, which is the dominant factor in the first iterations, to minimizing $y^T r$, which becomes dominant when the algorithm approaches convergence. This has the consequence that $f(u)$ becomes larger over the iterations. Now, if $f(u_{k+1}) > f(u_k)$, all values of the residual vector corresponding to the positive elements in y_{k+1} will decrease, as is demonstrated using the second relation in (4-14):

$$\begin{aligned} & f(u_{k+1}) - f(u_k) < y_{k+1}^T (r_k - r_{k+1}) \\ \text{if} \quad & f(u_{k+1}) - f(u_k) > 0 \\ \text{then} \quad & y_{k+1}^T (r_k - r_{k+1}) > 0 \end{aligned} \tag{4-21}$$

As all nonzero elements of vector y_{k+1} are mapped to zero due to the presence of only inequality constraints, the only way the last line of (4-21) is satisfied is when all elements of r corresponding to the positive elements of y_{k+1} decrease. Consequentially, as r equals the subgradient of y, the rate at which the dual vector increases will decrease for the elements of r which are greater than zero (the elements which violate the collision avoidance constraint). Then the same argument applies as before; if $y_{k+2} \to y_{k+1}$, also $r_{k+2} \to r_{k+1}$, which means that the constant stepsize ρ likely still results in $\Delta L > 0$.

The only moment when $f(u_{k+1})$ can become smaller than $f(u_k)$ is when elements in dual vector y become too large, causing overshoot in both $f(u)$ and Lagrangian function L. Therefore it is important to have graceful convergence towards $f(u^*)$ and $L(u^*, y^*)$.

Looking back at the residual defined in [21] for constant stepsize ρ given in (4-7), we see that the approximation $\|r_k\|_2 \leq S \ \forall \ k$ is too conservative for this problem: instead of an approximately constant r_k this value decreases over the iterations, i.e. $\sum_{i=1}^{k} \|r_i\|_2^2 < \infty$, which has the effect that the residual $\frac{\rho \sum_{i=1}^{k} \|r_i\|_2^2}{2k} \to 0$ for $k \to \infty$ also for constant stepsize ρ.

Convergence of the Lagrangian

In Figure 4-2 the consequence of a too large stepsize ρ can be seen. For $\rho = 3 \times 10^{-5}$, ΔL is always greater than zero which means that both the peaks and the valleys of the plot (corresponding to the Lagrangian function value after the y-update and the u-update respectively) increase monotonically. For $\rho = 5 \times 10^{-5}$, already in the third iteration (counter 5 to 7) $\Delta L < 0$, resulting in divergence of the algorithm. In Figure 4-3 it can be seen why the algorithm diverges for too large stepsize ρ. In iteration 6, the spacecraft pair 1-2 does not satisfy the collision constraint around sampling instance 5. Then, as ρ is chosen too large, some elements of dual vector y are increased too much. This causes the relative distance plot at iteration 7 to be almost opposite of the plot at iteration 6: spacecraft pair 1-2 is now collision-free around sampling instance 5, but some pairs which were safe before are pushed below the R_{col} bound. This means that the solutions are oscillating, preventing the algorithm from converging properly.

It is interesting to see that the Lagrangian value in Figure 4-1 starts to oscillate already around iteration 1750, whereas the primal residual continues to converges until iteration 3300. This can be explained by zooming in on the Lagrangian value over the iterations, as is shown in Figure 4-4. In the first iterations, the u-update decreases the Lagrangian value. When the algorithm approaches the optimal solution, the u-update cannot always minimize the

Lagrangian anymore, probably due to the limited numerical accuracy of the convex solver. What is important to notice however, is that the y-update is still increasing $y^T r$, and thus the weight is still shifting from minimizing $f(u)$ to minimizing $y^T r$. Therefore, despite the Lagrangian oscillating already from around iteration 1700, the primal residual and $\Delta\bar{V}$ are still converging until iteration 3300.

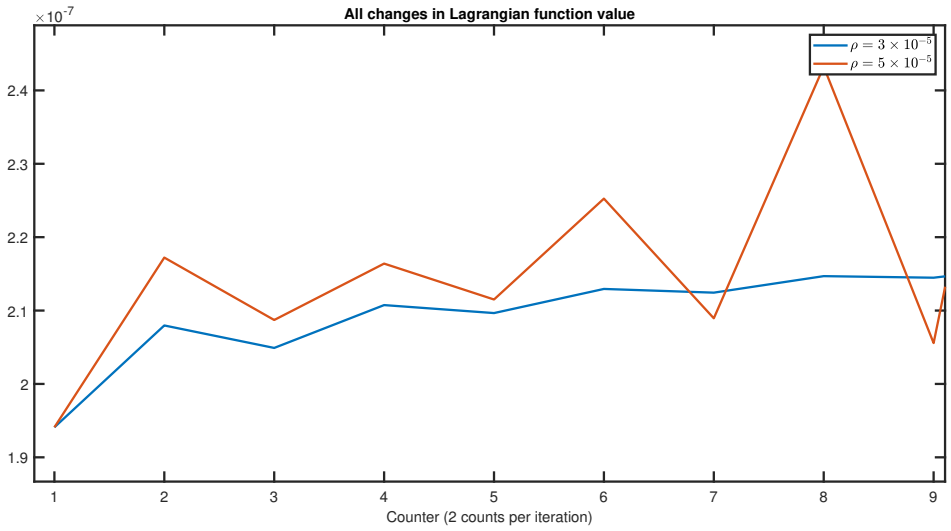


Figure 4-2: Lagrangian function value in stable versus unstable dual decomposition.

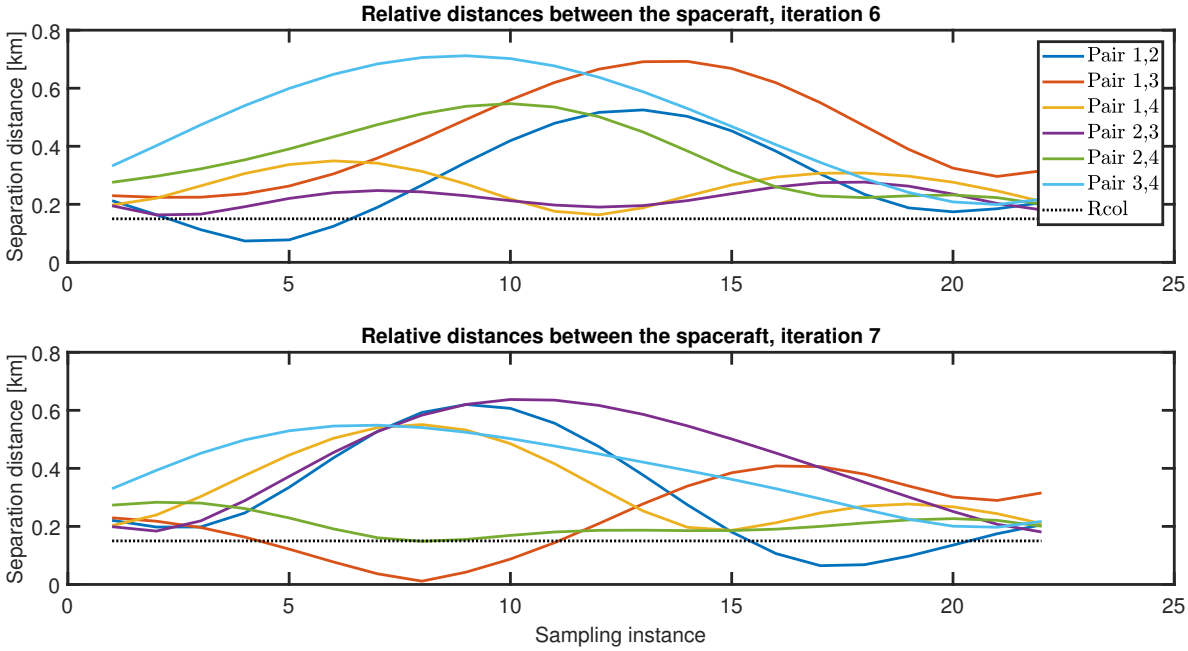
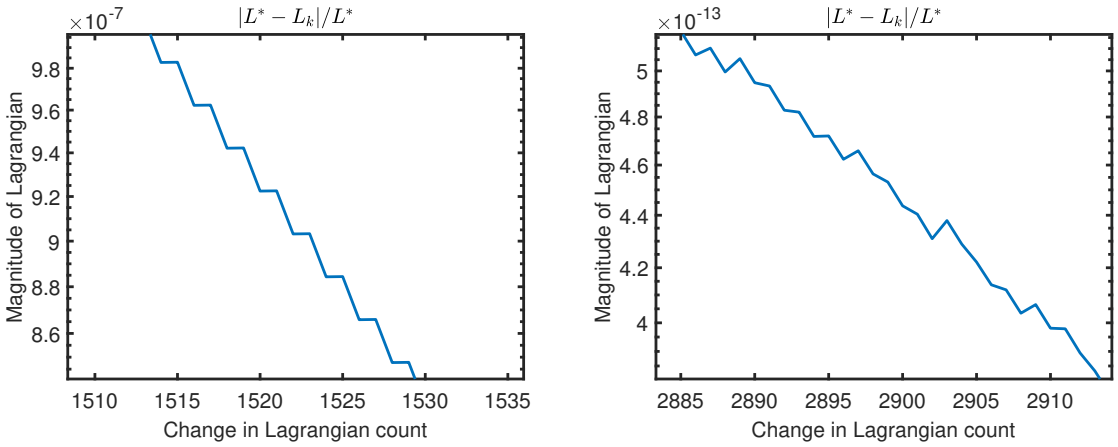


Figure 4-3: Unstable dual decomposition, relative distances between spacecraft pairs in iteration 6 versus iteration 7.



(a) Before convergence $|L^* - L_k|$ decreases after the y-update and increases after the u-update.

(b) Close to convergence $|L^* - L_k|$ decreases after the y-update but sometimes also decreases after the u-update.

Figure 4-4: Two instances of the scaled Lagrangian function value over the iterations. This function value changes twice per iteration: after the u-update and after the y-update. Close to convergence, the u-update in which the Lagrangian should be minimized does not always decrease the Lagrangian function value.

4-2 Augmented Lagrangian Methods

There are multiple methods to increase the convergence rate of dual decomposition based on ADMM. In ADMM instead of the Lagrangian the augmented Lagrangian is used, which includes an extra 2-norm of the included constraint. This can thus be seen as a robust version of dual decomposition. When the problem consists of more than two agents, a popular choice is to solve the problem with Consensus ADMM. This will be introduced first. Another way to solve such problem with more than two agents is the direct extension of ADMM. In general such algorithm is called an ALM. Both the Gauss-Seidel decomposition and the Jacobian decomposition of ALM are discussed as well, followed by a comparison of their convergence characteristics.

4-2-1 Alternating Direction Method of Multipliers in Consensus Form

If a problem consists of more than two agents, typically the consensus form of ADMM is used. In this form, consensus variables are introduced to decouple the problem [21]. Many authors have used such method to solve the collision free trajectory optimization problem [18, 19, 20, 42, 43]. It is based on the following idea. All agents calculate (approximate) solutions for (a subset of) the other agents. Using these estimations, local versions of the collision avoidance constraint can be defined. ADMM is then used to obtain consensus between the estimations and the actual optimization variables. The various papers differ in their exact implementation of this idea: it is still possible to choose exactly which variables are local and which are global, when and what information is shared and how each minimization problem is defined. One possibility is for example to let each agent estimate solutions for all agents, and to introduce a global consensus variable which is the average of the estimated solutions of the spacecraft [21].

As the specific problem setup by [20] can be clearly interpreted as a relaxed implementation of the centralized problem, this setup is recreated in this research. The problem which is solved using this implementation of Consensus ADMM is given in (4-22).

$$\begin{aligned} \min_{U_i,\ W_i,\ W_{j,i}\ \forall i,j} \quad & \frac{1}{2}\sum_{i=1}^{N} U_i^T U_i \\ \text{subject to} \quad & U_i \in \mathcal{U}_i,\ \forall i \\ & A_{ij}(W_{i,j} - W_i) - B_{ij} \leq 0,\ \forall\ i,j \\ & U_i = W_i = W_{j,i},\ \forall i,j \end{aligned} \tag{4-22}$$

Each agent makes a local copy of its own control vector which it denotes with W_i and constructs an estimation of the control vectors of the other spacecraft in its neighborhood, $W_{i,j},\ \forall j \in \mathcal{N}_i$. One spacecraft can now minimize its own control vector, while using the estimations of the trajectories in the collision avoidance constraints. If the last equality constraint of (4-22) is satisfied, the problem equals the convex problem given in (2-14). However, in Consensus ADMM this last equality constraint is included in the augmented Lagrangian, which means that the constraint will only be satisfied for $k \to \infty$.

The distributed Consensus ADMM algorithm is given for one spacecraft i in Algorithm 4, where $h(W_i, W_{i,j}) = A_{ij}(W_{i,j} - W_i) - B_{ij}$ is the collision avoidance constraint on the consensus

variable vectors W_i, $W_{i,j}$. It works as follows. The control vector U_i has to satisfy both separable constraints $U_i \in \mathcal{U}_i$, but neglects the collision avoidance constraint. In its update, it has to obtain consensus with both W_i (the collision free estimation made by spacecraft i itself) and $W_{j,i}$ (the collision free estimation of U_i made by spacecraft j). Consensus is obtained through the dual vectors y_i and $y_{j,i}$. In the next step, the consensus variables are updated. Vectors W_i and $W_{i,j}$ can neglect the separable constraints in their update, but have to be collision free. Consensus vector W_i has to obtain consensus with U_i using dual vector y_i, and the estimations $W_{i,j}$ have to obtain consensus with U_j using vector $y_{i,j,k}$.

It is shown in e.g. [20] that both consensus is achieved and an objective is minimized in a finite number of iterations. An advantage of this method is that each agent can update its control vector U_i in parallel. As all spacecraft construct a separate dual vector for each estimated control vector $W_{i,j}$, i.e. each $W_{i,j}$ has its own dual vector $y_{i,j}$, all spacecraft can update the dual vectors on their own in parallel and share them afterwards. Consensus ADMM is thus a fully distributed algorithm; all tasks are distributed over the entire formation.

As mentioned, this algorithm can be interpreted as a relaxed implementation of the centralized problem. Using this relaxation, the algorithm becomes a scalable alternative for larger formations. In step 3 of Algorithm 4, each spacecraft calculates its control vector without taking the collision avoidance constraint into account. In step 5 of Algorithm 4 essentially each spacecraft solves the entire centralized problem, but without the two separable constraints, i.e. the final configuration and maximum acceleration constraints. This largely simplifies the centralized problem. By dividing the constraints over two problems, both step 3 and step 5 become relatively computationally cheap, while the dual variable enforces consensus. ADMM reaches a modest accuracy relatively fast, but is known to converge very slowly to high accuracy [21]. In the following simulations it will be shown that this is indeed true for Consensus ADMM.

Algorithm 4 Consensus ADMM for spacecraft i

1: **Initialize** $W_0 = \mathbf{0}$, $y_1 = \mathbf{0}$, $k = 0$, A_{ij}, B_{ij}, $\forall j$
2: **repeat**
3: $k \leftarrow k + 1$
4: Update $U_{i,k}$ $\forall i = 1, \ldots, N$
$U_{i,k} \leftarrow \arg\min_{U_i} \frac{1}{2} U_i^T U_i + \frac{\rho}{2} \|U_i - W_{i,k-1} + \frac{y_{i,k}}{\rho}\|^2 + \sum_{j \in \mathcal{N}_i} \frac{\rho}{2} \|U_i - W_{j,i,k-1} + \frac{y_{j,i,k}}{\rho}\|^2$
s.t. $U_i \in \mathcal{U}_i$
5: Share U_i and receive U_j of neighbors $j \in \mathcal{N}_i$
6: Update W_i, $\{W_{i,j}\}_{j \in \mathcal{N}_i}$ $\forall i = 1, \ldots, N$
$W_{i,k}, \{W_{i,j,k}\}_{j \in \mathcal{N}_i} \leftarrow \underset{W_i, \{W_{i,j}\}_{j \in \mathcal{N}_i}}{\arg\min} \frac{\rho}{2} \|U_{i,k} - W_i + \frac{y_{i,k}}{\rho}\|^2 + \sum_{j \in \mathcal{N}_i} \frac{\rho}{2} \|U_{j,k} - W_{i,j} + \frac{y_{i,j,k}}{\rho}\|^2$
s.t. $A_{ij}(W_{i,j} - W_i) - B_{ij} \leq 0,\ \forall j \in \mathcal{N}_i$
7: Update $y_i, \{y_{i,j}\}_{j \in \mathcal{N}_i}$ $\forall i = 1, \ldots, N$
$y_{i,k+1} \leftarrow y_{i,k} + \rho(U_{i,k} - W_{i,k})$
$y_{i,j,k+1} \leftarrow y_{i,j,k} + \rho(U_{j,k} - W_{i,j,k})$ $\forall j \in \mathcal{N}$
8: Share $y_{i,k+1}$, $y_{i,j,k+1}$, receive $y_{j,k+1}$, $y_{j,i,k+1}$ from neighbors $j \in \mathcal{N}_i$
9: **until** Stopping criteria are met

Consensus ADMM Simulations

Three simulations of Consensus ADMM with varying stepsizes are shown in Figure 4-5. For a relatively small stepsize, $\rho = 3$, monotonous convergence of primal residual, Lagrangian and u can be seen. When ρ is increased slightly, the convergence rate is increased. This however comes at the cost of oscillations in the Lagrangian value, and consequently also oscillations in $\Delta\bar{V}$ and the primal and dual residuals. Increasing ρ even more decreases the convergence rate again, as the oscillations become too large. Interesting to see is that $\|U - W\|_2$, a measure for the consensus convergence, has to have an accuracy of approximately 1×10^{-10} in order to obtain a trajectory with $pr < 1 \times 10^{-6}$.

Now that the convergence characteristics of Consensus ADMM are more clear, the algorithm can be compared to the Augmented Lagrangian methods. First, these algorithms will be explained in the next sections.

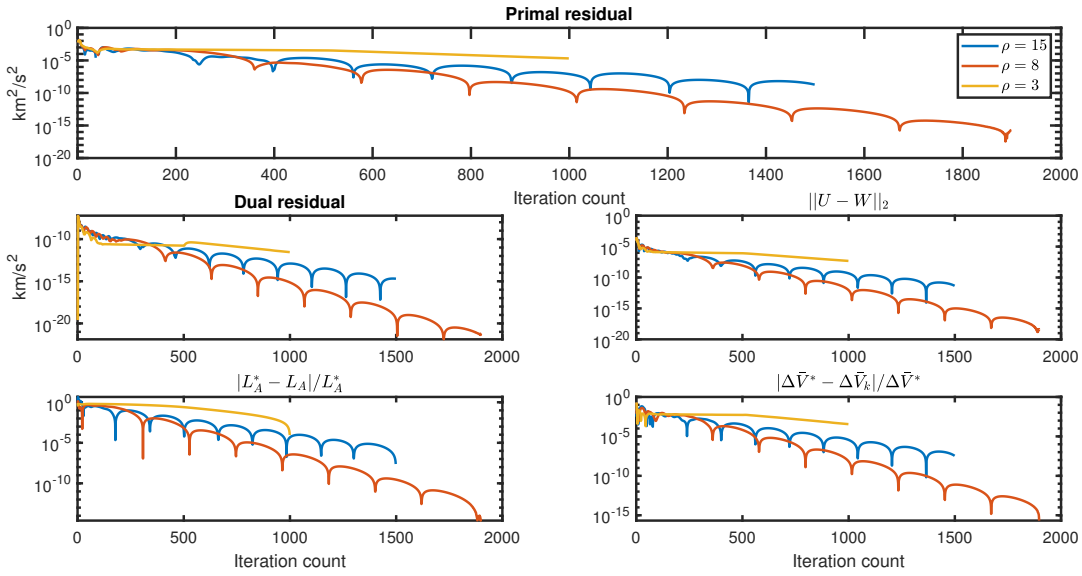


Figure 4-5: Consensus ADMM for 4 spacecraft with different stepsizes ρ.

4-2-2 Gauss-Seidel Decomposition of Augmented Lagrangian Method

Another method to solve the collision-free trajectory problem in a distributed way is to extend regular ADMM from two to N separable convex functions. The augmented Lagrangian then includes the collision avoidance constraint directly again, both in affine and quadratic expressions. Due to these quadratic expressions the problem is not decoupled anymore. A solution is to update the control acceleration vectors sequentially, using the most recent control vectors of the other spacecraft in the quadratic expression. This is called the GS-ALM. This means that a spacecraft receives a vector with the most recent control vectors of all spacecraft, updates its own part of this vector and shares the entire vector with the next spacecraft. The last spacecraft in the sequence first updates its part of the total control vector and then updates the dual vector. It shares both vectors again with the first spacecraft of the sequence

for the next iteration. This algorithm is again distributed, although the updating sequence has to be known to all spacecraft. A simple choice is Round Robin [44], which is also used in the GS-ALM simulations in this research.

The augmented Lagrangian for GS-ALM is shown in (4-23), where s is a slack variable vector which models the projection operator $[-]_+$: only positive elements should contribute in this two-norm. The augmented Lagrangian method with Gauss-Seidel decomposition is given in Algorithm 5. When $N > 2$, convergence of the algorithm is not guaranteed anymore, although the algorithm performs well in numerical simulations [45]. When the dual update is combined with a correction step based on dual variables at the previous iterate, i.e. a relaxation step, convergence can be guaranteed again [22]. In this research the algorithm is implemented without the relaxation step, as also for this specific problem the algorithm indeed converges properly. The fact that the control vectors are updated sequentially could lead to large waiting times for larger formations.

$$L_A(u, y, s) = \sum_{i=1}^{N} \frac{1}{2}\, U_i^T U_i + \frac{\rho}{2} \sum_{i=1}^{N-1} \sum_{j>i} \left\| A_{ij}(U_j - U_i) - B_{ij} + s_{ij} + \frac{y_{ij}}{\rho} \right\|_2^2 \tag{4-23}$$

Algorithm 5 Augmented Lagrangian method with Gauss-Seidel update scheme

Initialize $U_0 = \mathbf{0}, y_1 = \mathbf{0}$, $k = 0$, A_{ij}, B_{ij}, $\forall i, j$
repeat
  $k \leftarrow k + 1$
  Update U_i sequentially:
  for $i = 1, \ldots, N$ **do**
    Receive vector $U^T = [U_{1,k}^T, \ldots, U_{i-1,k}^T U_{i,k-1}^T, U_{i+1,k-1}^T, \ldots, U_{N,k-1}^T]^T$
      $U_{i,k} \leftarrow \arg\min_{U_i} L_{A,i}(U, y_k, s)$
    s.t. $U_i \in \mathcal{U}_i$
      $s \geq 0$
    Update $U_{i,k}$ in U and share U with neighbor $i + 1$
  Agent N updates y_{ij} $\forall i, j$
  $y_{ij,k+1} \leftarrow [y_{ij,k} + \rho(A_{ij}(U_{j,k} - U_{i,k}) - B_{ij})]_+$
  Shares y_{k+1} with agent 1
until Stopping criteria are met

GS-ALM Simulations

In contrast with dual decomposition, GS-ALM can converge with arbitrary stepsize ρ, as long as the solver is able to accurately solve the constrained minimization problem. The challenge is thus not to find the largest stable stepsize, but the one which results in the highest convergence rate. This is demonstrated in Figure 4-6, where the convergence is shown for different values of ρ. If the stepsize is chosen too small, the dual vector adjusts very slowly, and GS-ALM has similar convergence characteristics as dual decomposition: it monotonically increases $f(u)$ until $f(u) = f(u^*)$. If it is chosen too large, something interesting happens: the primal and dual residuals start to decrease quickly, but after a breaking point the convergence rate drops.

In the next section it will be explored why this would happen. Stepsize $\rho = 7 \times 10^{-4}$ seems to result in largest convergence rate with a monotonic decrease of the primal and dual residuals.

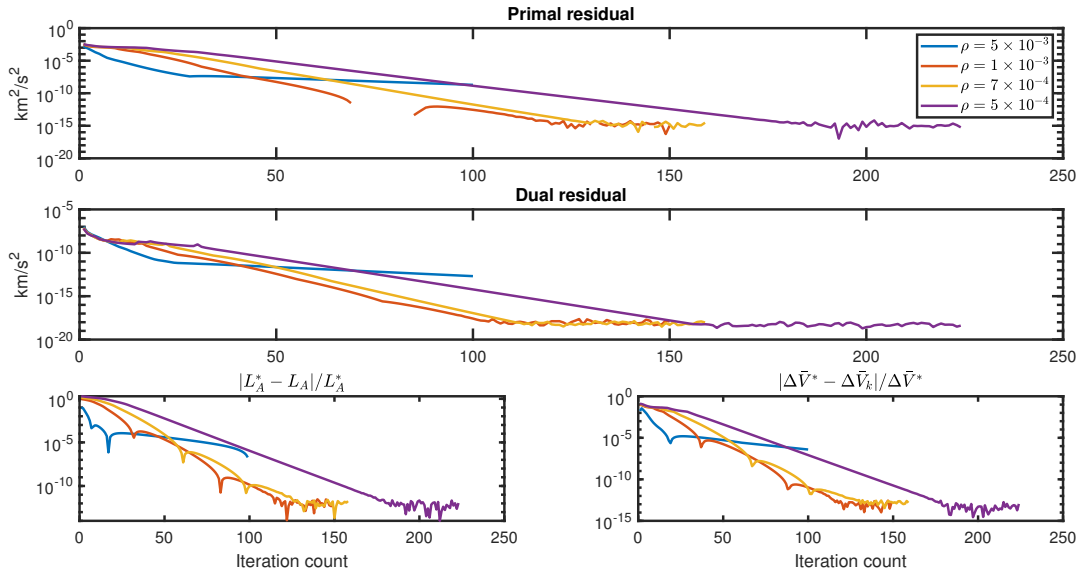


Figure 4-6: Convergence of GS-ALM with 4 spacecraft for different stepsizes ρ.

Choosing the GS-ALM Stepsize Too Large

In Figure 4-6 the plot corresponding to $\rho = 5 \times 10^{-3}$ behaves in an unexpected way: in the first few iterations it converges with a high rate, but after a breaking point, in this case around iteration 30, the convergence rate drops. The breaking point can be explained by looking at top plot in Figure 4-7, which shows all the positive elements of the residual vector for the simulation with $\rho = 5 \times 10^{-3}$. The primal residual is defined to be the maximum of this plot. As can be seen, at iteration 7 and 28 different elements become largest, explaining the sudden changes in convergence rate of the primal residual plot. This in contrast to the lower plot in Figure 4-7, which shows the positive residuals for $\rho = 7 \times 10^{-4}$, which decrease monotonically.

The fact that the convergence rate decreases also on the logarithmic scale has to do with the minimization of L_A. The residual r depends on the balance between minimizing the left hand side of L_A, i.e. $\Delta\bar{V}$, and minimizing the right hand side, i.e. the residual multiplied by the dual vector. If ρ is selected small enough, the dual vector updates only slowly and this balance shifts gradually. As $\Delta\bar{V}$ approaches $\Delta\bar{V}^*$ from below, updates in the dual vector have a relatively large weight. The result is a monotonically increasing left hand side and a monotonically decreasing right hand side. If ρ is chosen too large however, the dual is updated with larger steps which results in overshoot. The left hand side of L_A now becomes too large. The dual update compensates for this, but as $\Delta\bar{V}$ approaches $\Delta\bar{V}^*$ from above, the updates in the dual vector have a relatively small weight. This results in slower adjustment and a lower convergence rate. In Figure 4-8 it is demonstrated that for $\rho = 5 \times 10^{-3}$, $\Delta\bar{V}_k$

overshoots and then approaches $\Delta\bar{V}^*$ from above. Note that in iteration 15 $\Delta\bar{V}$ is already relatively close to $\Delta\bar{V}^*$. The plot with $\rho = 7 \times 10^{-4}$ on the other hand approaches $\Delta\bar{V}^*$ from below at a much lower rate. A large stepsize thus results in modest accuracy relatively fast, but when it overshoots the convergence rate drops resulting in slow convergence to high accuracy.

Another way to show that the stepsize is chosen too large is by adjusting the algorithm slightly: instead of always minimizing the control acceleration vectors in the same sequential order (from spacecraft 1 to 4), the order can also be randomized. The result of this is shown in Figure 4-9. For $\rho = 5 \times 10^{-3}$, the spacecraft which is first in the update sequence always overcompensates, resulting in an inaccurate dual vector update and oscillations in the residuals and Lagrangian. If the stepsize is chosen small enough, also the first spacecraft updates properly, resulting in accurate dual updates, less oscillations and faster convergence.

Although GS-ALM shows good convergence properties, the sequential updates of the control vectors result in large waiting times for individual spacecraft. Especially in larger formations the time required to run the algorithm one iteration will increase as all spacecraft have to wait for each other. This can be solved by using a Jacobian decomposition instead of the Gauss-Seidel decomposition, as will be explained in the next section.

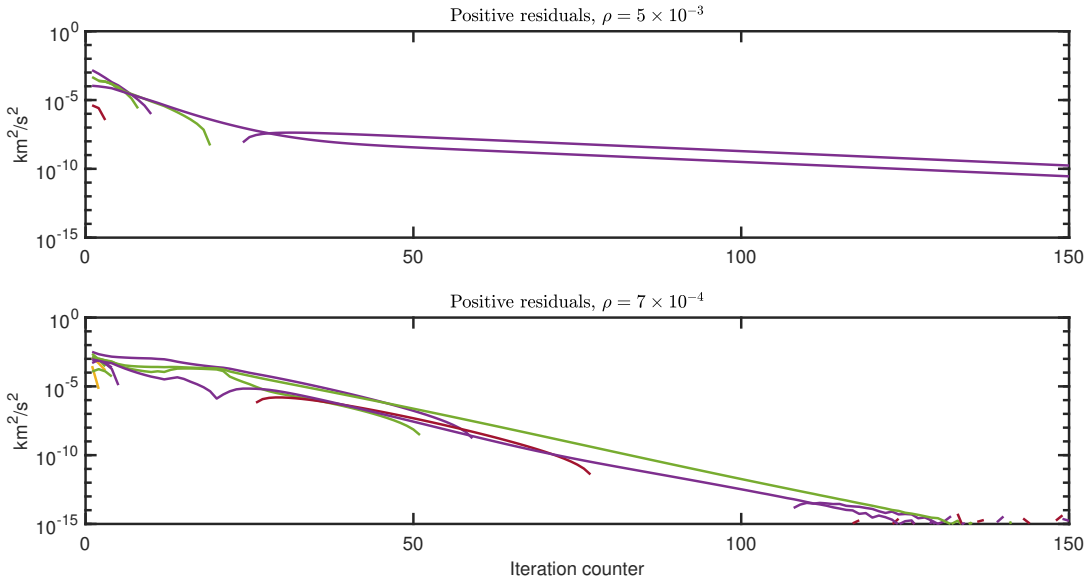


Figure 4-7: Convergence of all residual elements: most of them are negative and thus collision-free, only few are positive. Top figure: when ρ is chosen too large, different residual elements become largest over the iterations. Lower figure: when ρ is chosen small enough, the largest residual element converges almost monotonically.

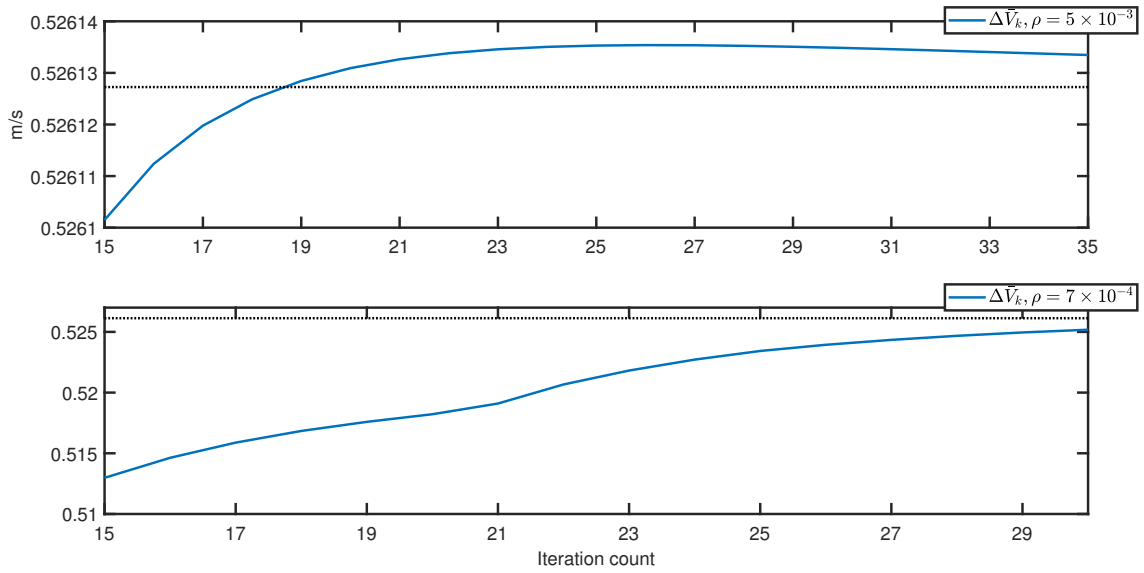


Figure 4-8: The average required velocity change over the iterations for $\rho = 5 \times 10^{-3}$ versus $\rho = 7 \times 10^{-4}$. Note the y-scales: the top figure is much closer to $\Delta\bar{V}^*$ than the lower figure.

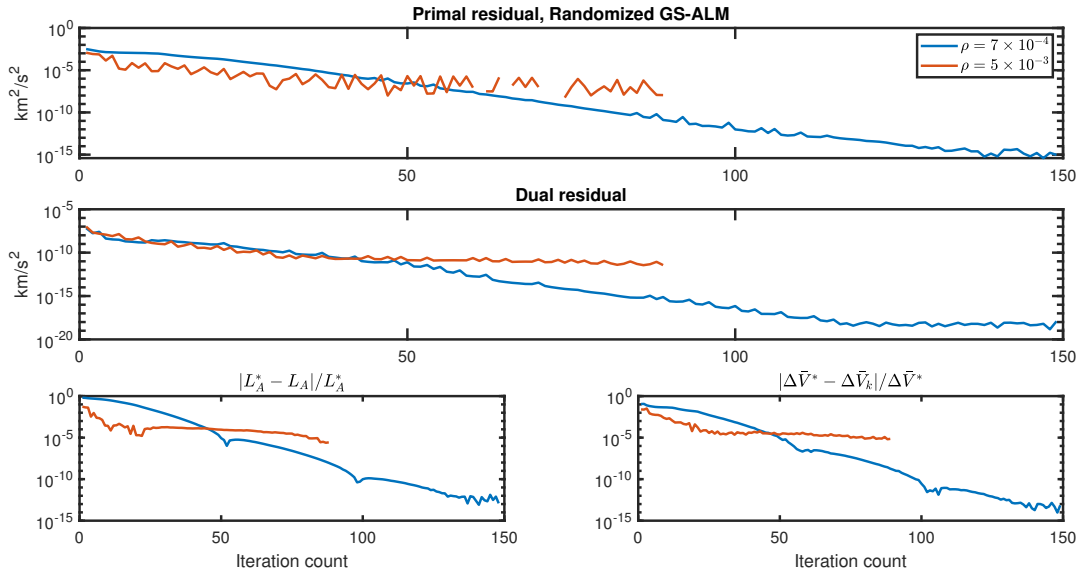


Figure 4-9: Convergence of GS-ALM: 4 spacecraft with random spacecraft update order. The properly chosen stepsize still results in convergence, whereas the stepsize chosen too large results in oscillations due to overcompensation of the spacecraft which updates first.

4-2-3 Jacobian Decomposition of Augmented Lagrangian Method

In the Jacobian decomposition algorithm, all spacecraft update their control vectors in parallel, instead of the sequential Gauss-Seidel decomposition. This algorithm then requires a master node to update the dual variable. There are ways to distribute this task as well. As the coupled collision avoidance constraint, and thus also the corresponding dual vector, is defined for spacecraft pairs, it is possible to let one of the two spacecraft in each pair update the dual vector. Next to this, each spacecraft should share its latest obtained control acceleration vector with all other spacecraft. With these modifications, again all tasks are distributed over the formation. Such distribution scheme is demonstrated for example in [17]. The J-ALM algorithm is given in Algorithm 6.

When ALM is altered using the Jacobi scheme, the standard convergence proofs do not hold anymore, even for $N = 2$ [16]. To guarantee convergence again, a relaxation step is introduced [16]. Instead of using the output of one dual iteration directly as the input for the next iteration, the output is combined with a relaxation step, as is shown in step 10 of Algorithm 6. In [16] it is shown that an appropriately chosen relaxation parameter can ensure strict contraction of the iterations. It is proven that J-ALM will converge if relaxation parameter ν is chosen to be $\nu \in (0, 2(1-\sqrt{\frac{N}{N+1}}))$, which means for $N = 4$ that $\nu \in (0, 0.2111)$. This result was however obtained for a problem with coupling in only equality constraints. It is likely that with inequality instead of equality constraints in the Lagrangian this relaxation parameter can be chosen less conservative, just like the stepsize in dual decomposition as discussed in Section 4-1-1.

Algorithm 6 Augmented Lagrangian method with Jacobi update scheme

1: **Initialize** $U_0 = \mathbf{0}, y_1 = \mathbf{0}$, $k = 0$, A_{ij}, B_{ij}, $\forall i, j$
2: **repeat**
3: $k \leftarrow k + 1$
4: Central agent shares U_{k-1} and y_k with all spacecraft $i = 1, \ldots, N$
5: Update U_i in parallel
6: **for** $i = 1, \ldots, N$ **do**
7: Receive vector $U^T = [U_{1,k-1}^T, \ldots, U_{i-1,k-1}^T U_{i,k-1}^T, U_{i+1,k-1}^T, \ldots, U_{N,k-1}^T]^T$
8: $\hat{U}_{i,k} \leftarrow \arg\min_{U_i} L_{A,i}(U, y_k, s)$
s.t. $U_i \in \mathcal{U}_i$
$s \geq 0$
9: Return $\hat{U}_{i,k}$ to central agent
10: Central agent updates y_{ij} $\forall i = 1, \ldots, N-1$, $j > i$ and U
$\hat{y}_{ij,k+1} \leftarrow y_{ij,k} + \rho(A_{ij}(\hat{U}_{j,k} - \hat{U}_{i,k}) - B_{ij})$
$y_{ij,k+1} \leftarrow [y_{ij,k} - \nu(y_{ij,k} - \hat{y}_{ij,k+1})]_+$
$U_k \leftarrow U_{k-1} - \nu(U_{k-1} - \hat{U}_k)$
11: **until** Stopping criteria are met

J-ALM Simulations

Again the algorithm is simulated with varying stepsizes and relaxation parameters. The result is shown in Figure 4-10. For relatively large $\rho = 3 \times 10^{-3}$ and $\nu = 0.5$ convergence is again fast to modest accuracy, but slower afterwards. The stepsize which was optimal for GS-ALM, $\rho = 7 \times 10^{-4}$, now results in overshoot in the Lagrangian and a jump of the primal residual to zero between iteration 150 and 225. Decreasing ρ even further with $\nu = 0.5$ results in linear convergence, but at a slower rate. In the fourth and fifth lines of Figure 4-10 the influence of relaxation parameter ν is demonstrated: for $\rho = 4 \times 10^{-4}$ and $\nu = 0.5$ the algorithm converges linearly, but for $\nu = 0.7$ the algorithm starts to oscillate as the subgradients obtained in a parallel way are not entirely accurate. Lowering ν to 0.2 results in relatively slow convergence.

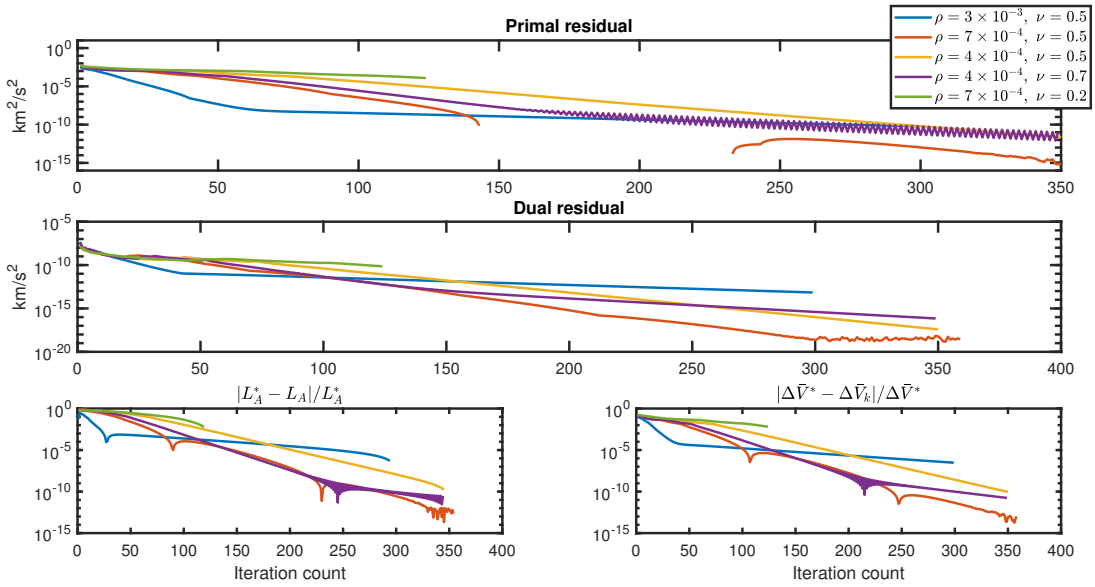


Figure 4-10: Convergence of J-ALM with 4 spacecraft for different stepsizes ρ and relaxation parameters ν. The red line with $\rho = 7 \times 10^{-4}$ and $\nu = 0.5$ converges fastest, although the primal residual is zero between iteration 145 and 235. Decreasing ν to 0.2 results in much slower convergence, increasing it to 0.7 results in undesired oscillations.

Just like in GS-ALM, choosing ρ too large results in fast convergence to moderate accuracy, and slow convergence to high accuracy, as is shown in Figure 4-10. However, as in J-ALM all spacecraft update their control acceleration vector in parallel, it is not just one spacecraft which overcompensates: all updates are not entirely accurate. This increases the effect on the convergence rate discussed in Section 4-2-2.

4-2-4 Simulation and Comparison

In the previous sections, already some simulations are shown to give insight in the convergence characteristics of the algorithms for different parameter values. To make sure that the previous simulations are a good indication of general convergence characteristics, the three algorithms are simulated on two more reconfiguration scenarios. The convergence iteration is recorded at which both residuals met the stopping criteria $pr \leq 1 \times 10^{-6}$ and $dr \leq 1 \times 10^{-11}$. For

all algorithms ρ is tuned to balance rise time and overshoot of the Lagrangian, i.e. fast but smooth convergence. Three to four different stepsizes ρ have been simulated, the fastest result is recorded.

The convergence results of the three algorithms averaged over three reconfiguration scenarios are given in Table 4-1. For different scenarios, all algorithms required new tuning of the stepsize, a time-consuming process. Although Consenus ADMM requires the lowest computational time per iteration, both ALM algorithms converge in much less iterations on average.

	GS-ALM	J-ALM	Consensus ADMM
Av. Convergence iter.	36.3	70.7	278
Av. comp time per iter. [s]	3.12	2.47	1.67

Table 4-1: Convergence characteristics for the three different algorithms on 4 spacecraft reconfigurations averaged over 3 simulations.

The ALM algorithms clearly outperform this specific implementation of the consensus ADMM algorithm for the generation of collision free trajectories. This difference is likely due to two reasons. The first reason is that the ALM makes use of an inequality constraint in the augmented Lagrangian, whereas the Consensus ADMM uses an equality constraint. Using the inequality constraint, the dual variable has to influence the control vector only at the parts which are not yet collision free. However, with the equality constraint each element of the control vector has to be influenced to reach consensus. All dual variables have to be balanced very accurately, which requires much more iterations. Compare for example Figures 4-5 and 4-6: the consensus ADMM plot has many oscillations in primal and dual residual, Lagrangian and cost, indicating oscillations in the dual vector as well. GS-ALM has much less oscillations, with primal and dual residual converging monotonically for properly chosen stepsize. This also relates to the discussion about dual decomposition with coupling in the inequality constraints which can converge even with constant stepsize.

The second reason is the required accuracy of the algorithm. The Lagrangian of the ALM algorithms acts on the collision avoidance constraint directly. If the algorithm reaches a constraint satisfaction accuracy of 1×10^{-6}, this translates directly to the stopping criteria on pr. However, the Lagrangian of the consensus ADMM algorithm includes the equality constraints between optimization variables and consensus variables. It only meets the stopping criteria on pr when consensus is obtained up to a constraint satisfaction accuracy of 1×10^{-10}, as can be seen in for example Figure 4-5. Consensus ADMM thus has to converge to an accuracy 4 orders of magnitude higher than the ALM algorithms to meet the stopping criteria.

As the ALM methods can converge more predictable and are able to find a solution with modest accuracy more quickly than this implementation of the Consensus ADMM, it is concluded that the ALM methods are better suited to solve the collision free trajectory optimization problem. As J-ALM scales better with the size of the formation than GS-ALM without significant loss of convergence rate, this algorithm seems most promising. Whether other implementations of Consensus ADMM yield similar performance is a question which could be looked into in future work.

4-3 Asynchronous Algorithm

When using J-ALM to optimize the reconfiguration trajectories, the central node waits for all spacecraft to return their updated control vectors before the dual vector is updated. Especially for larger formations where some spacecraft may require more time to update the control vectors (they could have more neighbors and thus a larger problem for example) or where a varying communication delay is present, this may again result in long idle times of the individual spacecraft. Therefore it is interesting to look at an asynchronous implementation of J-ALM. The asynchronous algorithms for respectively the central node and one agent in the network are shown in Algorithms 7 and 8. The tasks of the central node can be distributed in the same way which is discussed for synchronous J-ALM, although now all spacecraft should hold also the previous control acceleration vectors of neighbors in memory to use in case they did not yet return the most recent control vector.

Algorithm 7 Central node

Initialize $U_0 = \mathbf{0}, y_1 = \mathbf{0}$, $k = 0$, $A_{ij},\ B_{ij},\ \forall i, j$
repeat
  $k \leftarrow k + 1$
  Send all spacecraft $i \in N$ the dual vector y, the most recent control sequences $U_i\ \forall\ i = 1, \ldots, N$ and a request for an update of U_i
  Wait for a set $\mathcal{W}$ of spacecraft to return their updated control vectors
  for $i = 1, \ldots, N$ **do**
    if $i \in \mathcal{W}$ **then**
      Update $U_{i,k}$
    else
      $U_{i,k} \leftarrow U_{i,k-1}$
  Update subgradients $r_{ij,k} = A_{ij}(U_{j,k} - U_{i,k}) - B_{ij},\ \forall i, j$
  Update dual vector and control vectors:
  $\hat{y}_{k+1} \leftarrow y_k + \rho r_k$
  $y_{k+1} \leftarrow [y_k - \nu(y_k - \hat{y}_{k+1})]_+$
  $U_k \leftarrow U_{k-1} - \nu(U_{k-1} - \hat{U}_k)$
  Send spacecraft $i \in W$ vectors y_{k+1} and $U_{i,k}\ \forall\ i = 1, \ldots, N$, request new control vectors $U_{i,k+1}, \forall i \in W$
until Convergence criteria are met

Algorithm 8 Agent

Receive y and $U_i\ \forall\ i = 1, \ldots, N$
Update U_i
Send updated control vector to central node with certain delay

Asynchronous convergence results for incremental gradient methods are obtained before for large scale distributed optimization in the paper by Aytekin and Johansson [23]. However, their results require a strongly convex dual problem. Asynchronous dual decomposition is discussed in [24]. In their proof of the convergence characteristics, explicit solutions of the minimization of the augmented Lagrangian are required. These are however not available for

the collision free trajectory optimization problem. Therefore, proving asynchronous convergence for the trajectory optimization problem is not trivial.

An intuitive approach to implementing J-ALM in an asynchronous way, is to divide a suitable stepsize for synchronous J-ALM trough the maximum delay present in the asynchronous setting, $T_{\max}$. Here $T_{\max}$ is defined to be a delay of a certain number of iterations. This means that the maximum delay is defined relatively to other spacecraft: if $T_{\max} = 5$ this means that in 5 iterations each spacecraft has returned a control vector at least once. If for small changes in the dual vector the residual changes linearly, the total dual vector update in the asynchronous algorithm after $T_{\max}$ iterations with $\rho/T_{\max}$ would be more accurate than the update in the synchronous algorithm with ρ in 1 iteration. This is because the control vector update of the agent with $T_{\max}$ delay is used $T_{\max}$ times, and thus has the same effect on the dual update as one update in the synchronous algorithm, whereas the spacecraft which update more frequently actually result in more accurate updates of the dual vector than the synchronous algorithm. If this approach would be valid, then convergence is expected within $T_{\max}$ times the convergence iteration of J-ALM; i.e. compare for example with J-ALM (see Figure 4-10): for $T_{\max} = 5$, $\rho = \frac{7\times10^{-4}}{5} = 1.4 \times 10^{-4}$, convergence would be expected before iteration $5 \times 350 = 1750$.

This simulation is performed for four spacecraft with a random delay T, $T_{\max} = 5$ iterations, $\rho = 1.4 \times 10^{-4}$, $\nu = 0.5$, and the central node waits for 2 updated control vectors before it updates the dual vector. In the simulation, each iteration 2 agents are chosen using an uniform distribution, unless an agent did not update its control vector for four iterations; then this agent is chosen to return its control vector update in the next iteration. The result is shown in the blue line in Figure 4-11. The algorithm does converge properly, with only small oscillations. This indicates that for small changes in the dual vector, we can expect small changes in the residual as well, and if ρ is small enough than we can still expect convergence of the asynchronous algorithm. In the first few hundreds of iterations, the dual residual regularly jumps to $O(1 \times 10^{-18})$. This happens when two spacecraft which are both already collision free return their control vector updates. Those trajectories are not influenced by the updated dual vector, so they do not change. The case when ρ is not decreased properly is shown in the red line in Figure 4-11. Here, the stepsize is not divided by the maximum delay, so $\rho = 7 \times 10^{-4}$ and $\nu = 0.5$, for $T_{\max} = 5$. It is clear that ρ is too large, the algorithm is not able to converge further than $pr \approx 1 \times 10^{-6}$.

Finally, in Figure 4-12 it is shown that decreasing ρ proportional to $T_{\max}$ also works for larger maximum delays. Here, $T_{\max} = 15$, $\rho = \frac{7\times10^{-4}}{15}$, $\nu = 0.5$ and the central node waits for 2 control vector updates before it updates the dual vector. Clearly, both the primal residual and the control acceleration vector converge smoothly again, and convergence is obtained around iteration 4000, which is smaller than $T_{\max} \times 350 = 5250$.

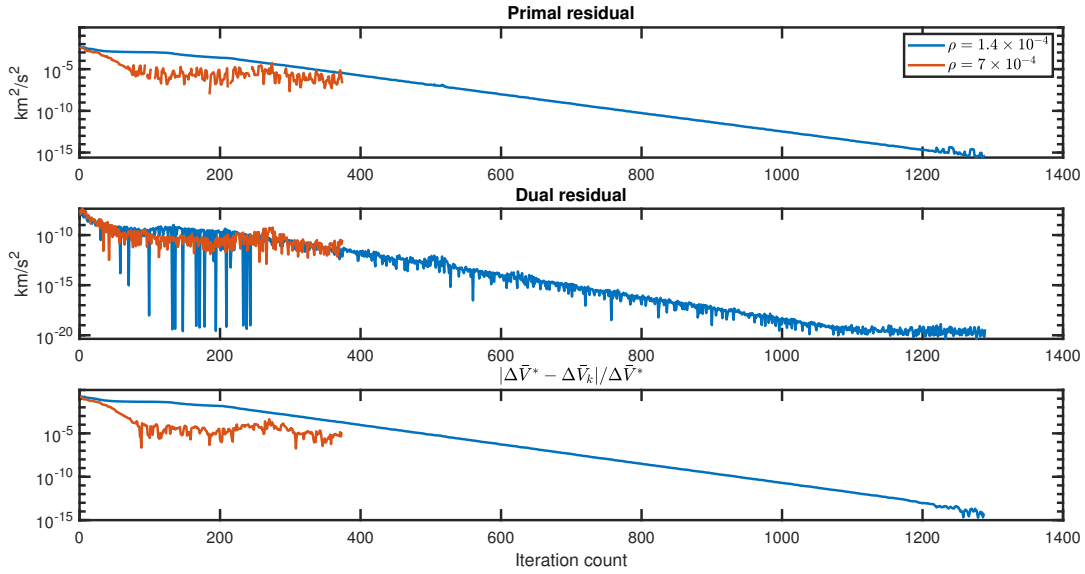


Figure 4-11: Asynchronous J-ALM, waiting for 2 control vector updates per iteration, random delay with $T_{\text{max}} = 5$ and $\nu = 0.5$. The stepsize suitable for synchronous J-ALM results in oscillations in the asynchronous case, but dividing the stepsize through the maximum delay results in convergence.

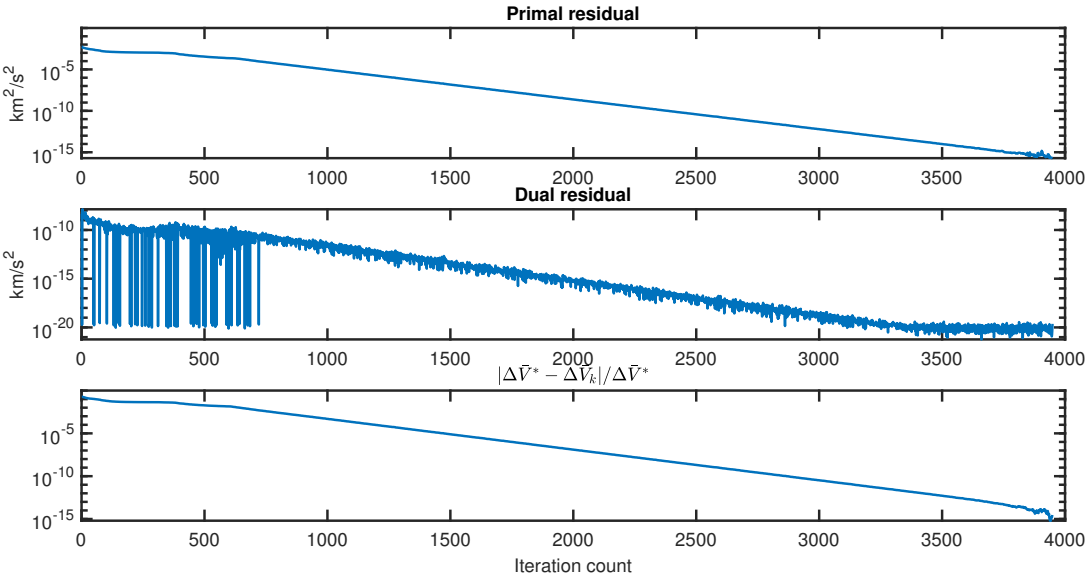


Figure 4-12: Asynchronous J-ALM, waiting for 2 control vector updates per iteration, random delay with $T_{\text{max}} = 15$, $\rho = (7/15) \times 10^{-4}$ and $\nu = 0.5$. Also for larger delays convergence can be obtained when the stepsize is adjusted appropriately.

4-4 Summary and Conclusions

In this chapter various dual algorithms are compared on their convergence rate to solve the collision free trajectory optimization problem, respectively dual decomposition, Consensus ADMM, GS-ALM, synchronous J-ALM and asynchronous J-ALM. All algorithms converge to the same solution as a centralized algorithm. Dual decomposition requires an order of magnitude more iterations to converge than the ADMM and synchronous ALM methods. Synchronous GS- and J-ALM also outperformed Consensus ADMM: although they required more time per iteration they converged in much less iterations. As J-ALM is also better scalable than GS-ALM due to the parallel instead of sequential update of the gradients, J-ALM seems most promising to solve the collision-free trajectory optimization problem at this point.

By simulation it is shown that J-ALM can also converge in an asynchronous setting, when the stepsize suitable for synchronous J-ALM is divided by the maximum delay. This potentially decreases the time required to solve the problem, as the idle time of spacecraft can be decreased.

Before the algorithms are simulated in the SCP framework, an effort is made to accelerate the convergence rates. Various acceleration methods are known to accelerate the convergence of gradient methods. These acceleration methods will be discussed and applied to dual decomposition, synchronous and asynchronous J-ALM in the next chapter.

Chapter 5

Accelerated Dual Methods with Sequential Convex Programming

In this chapter first the acceleration of dual decomposition and the synchronous and asynchronous implementations of J-ALM are discussed. Again all simulations are performed using the same four spacecraft reconfiguration. Then, the accelerated algorithms are simulated with SCP, to see how SCP influences the convergence rate and to be able to compare the dual methods to the hybrid cooperation algorithm.

5-1 Accelerating Dual Algorithms

Many acceleration techniques have been proposed to increase the convergence rate of gradient methods. Most of those acceleration techniques use memory of previous iterates in order to improve the accuracy of the next step direction. In this research, three acceleration methods will be discussed: HBA, FISTA and AA. First the methods will be explained, and some simulations are shown to explain the effect of varying the acceleration parameters. Then all acceleration methods will be simulated on dual decomposition, synchronous J-ALM and asynchronous J-ALM to be able to compare the different acceleration methods with each other.

5-1-1 Heavy Ball Acceleration

The HBA is the most simple way to include memory, or a momentum term, in the dual update. The new search direction is based on a combination of the new subgradients and the old search direction [25]. For dual decomposition, the update is given as follows, with momentum parameter $\beta > 0$:

$$y_{ij,k+1} = [y_{ij,k} + \rho r_{ij,k} + \beta(y_{ij,k} - y_{ij,k-1})]_+ \tag{5-1}$$

A simulation of HBA implemented on dual decomposition is shown in Figure 5-1. The convergence rate can be significantly increased. For $\beta = 0.7$, the primal residual convergence plot starts to oscillate a little in the first 25 iterations and for $\beta = 0.9$ the convergence is fastest but with many oscillations.

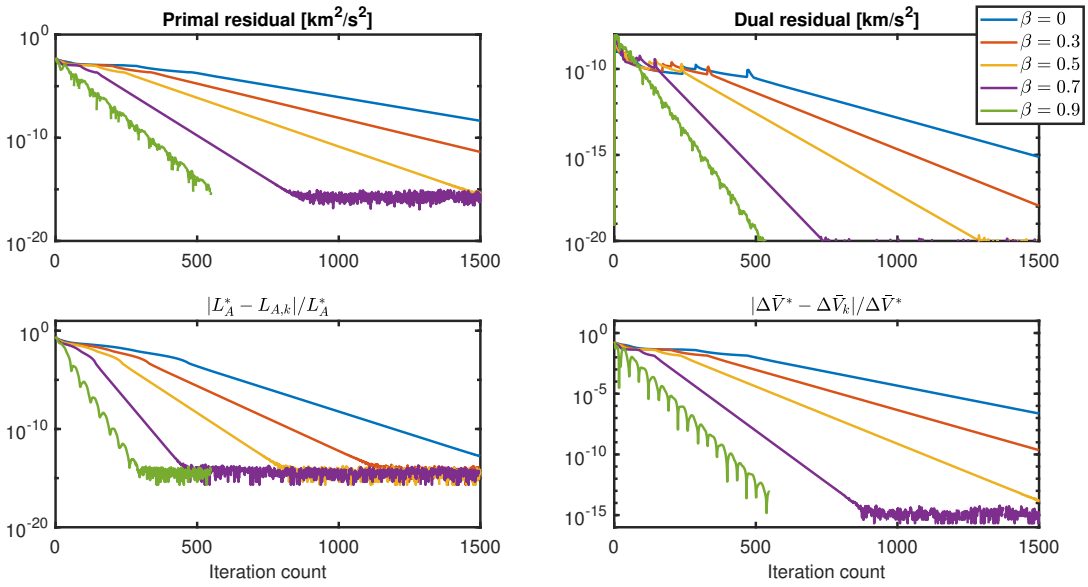


Figure 5-1: Heavy Ball on dual decomposition, with $\rho = 3 \times 10^{-5}$ and different values for momentum parameter β.

For the synchronous and asynchronous J-ALM algorithms the Heavy Ball dual update looks similar as for dual decomposition; it only includes the extra relaxation parameter:

$$\begin{aligned} \hat{y}_{ij,k} &= y_{ij,k} + \rho r_{ij,k} \\ y_{ij,k+1} &= [y_{ij,k} + \nu(\hat{y}_{ij,k} - y_{ij,k}) + \beta(y_{ij,k} - y_{ij,k-1})]_+ \end{aligned} \tag{5-2}$$

However, whereas in dual decomposition each spacecraft minimizes its own control acceleration vector without taking into account the control vectors of other spacecraft, in J-ALM the augmented Lagrangian also includes fixed control acceleration vectors from other spacecraft. If only the dual vector is updated with Heavy Ball momentum term, there is a mismatch between the dual vector and the fixed control acceleration vectors. Instead of the regular J-ALM u-update as given in Algorithm 6, it can also be updated as follows:

$$U_k \leftarrow U_{k-1} - \nu(U_{k-1} - \hat{U}_k) + \beta(U_{k-1} - U_{k-2}) \tag{5-3}$$

In Figure 5-2, HBA on synchronous J-ALM is simulated, both with the regular u-update and with the Heavy Ball u-update given in (5-3). It is clear that both methods accelerate the convergence, but adjusting U together with the dual vector has a much larger effect; it converges approximately twice as fast.

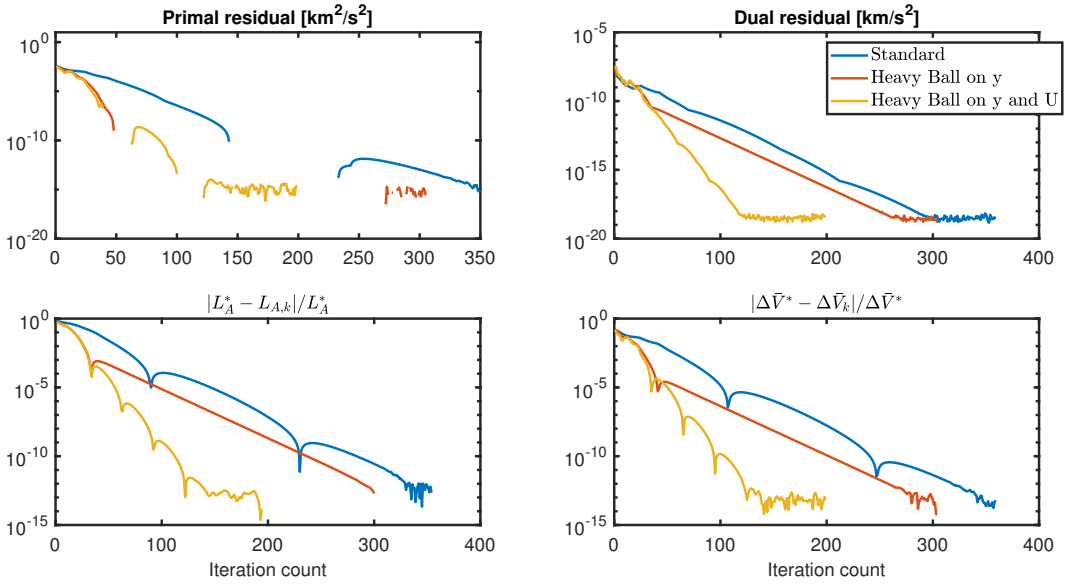


Figure 5-2: Heavy Ball on synchronous J-ALM with $\rho = 7 \times 10^{-4}$, $\nu = 0.5$ and $\beta = 0.5$. Updating U in the same way as dual vector y has a large influence on the convergence rate.

The HBA on asynchronous J-ALM is more difficult, as both y_k as y_{k-1} are obtained with partly outdated control vectors and thus not entirely accurate. Possibly the same reasoning can be used as with the selection of a proper the stepsize: if the maximum delay is for example 5 iterations, then β could be chosen to be $0.5/5 = 0.1$. This way, the control vector of the agent which suffers maximum delay will contribute the same to the dual vector update as in the synchronous case with $\beta = 0.5$, whereas control vectors which are updated more frequently actually increase the accuracy. As J-ALM in combination with HBA was accelerated significantly with $\rho = 7 \times 10^{-4}$, $\nu = 0.5$ and $\beta = 0.5$, the asynchronous algorithm with $T_{\max} = 5$ would then potentially be accelerated with $\rho = (7/5) \times 10^{-4}$, $\nu = 0.5$ and $\beta = 0.1$. The result of this simulation is shown in Figure 5-3. Again U is updated using the same momentum update rule as y.

For $\beta = 0.1$ indeed a small acceleration is obtained, resulting in convergence around iteration 1200 instead of 1300. For $\beta = 0.5$ the algorithm does not converge: the momentum parameter is too large resulting in oscillating solutions. The simulation with $\beta = 0.3$ shows that heavy ball acceleration can increase the convergence rate, but this specific value is found by trial and error, without proper justification.

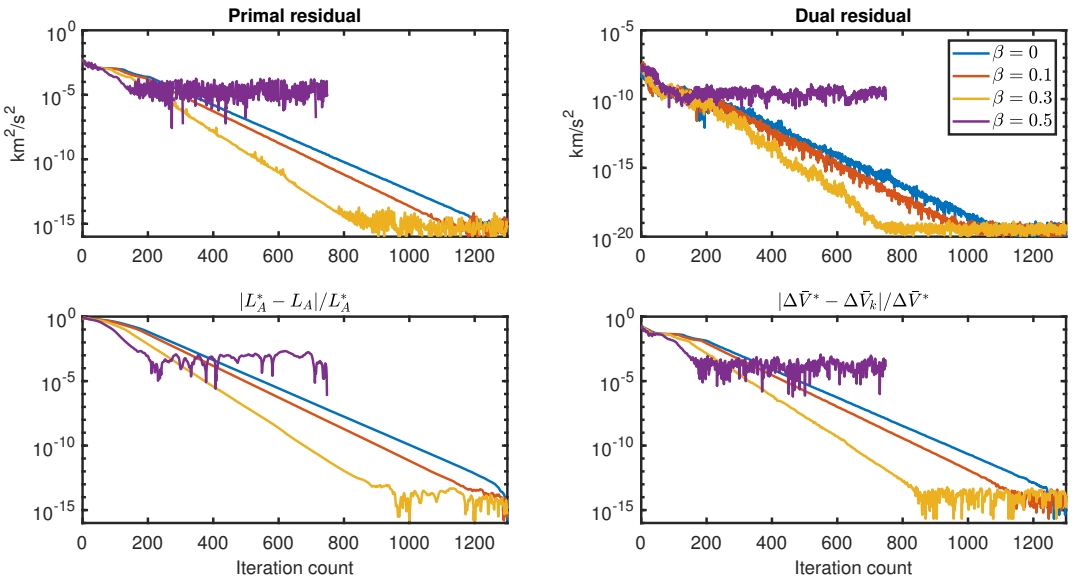


Figure 5-3: Heavy Ball on asynchronous J-ALM with $\rho = 1.4 \times 10^{-4}$, $\nu = 0.5$, $T_{\text{max}} = 5$ and a central node waiting for 2 control vectors. Momentum parameter $\beta = 0.1$ results in little acceleration whereas $\beta = 0.5$ does not converge. The algorithm can be more significantly accelerated with $\beta = 0.3$.

5-1-2 Fast Iterative Shrinkage-Threshold Algorithm

A method similar to HBA but with improved worst-case convergence bound is FISTA, an acceleration method based on Nesterov acceleration [26]. It improves the convergence rate bound from $O(1/k)$ to $O(1/k^2)$. The algorithm as given in [27] is shown in Algorithm 9. As this version of FISTA is proposed by Beck and Teboulle, it is called FISTA-BT.

Algorithm 9 FISTA-BT on dual decomposition

1: **Initialize** $y_1 = \mathbf{0}$, $\hat{y}_0 = \mathbf{0}$, $k = 0$, $\tau_1 = 1$, $A_{ij}, B_{ij}, \ \forall i, j$
2: **repeat**
3: $\quad k \leftarrow k + 1$
4: $\quad$ Update U_i in parallel:
5: $\quad$ **for** $i = 1, \ldots, N$ **do**
6: $\quad\quad U_{i,k} \leftarrow \underset{U_i}{\text{argmin}}\ L_i(u, y_k)$ s.t. $U_i \in \mathcal{U}_i$
7: $\quad \hat{y}_k \leftarrow y_k + \rho r_k$
8: $\quad \tau_{k+1} \leftarrow \frac{1+\sqrt{1+4\tau_k^2}}{2}$
9: $\quad y_{k+1} \leftarrow [\hat{y}_k + \frac{\tau_k - 1}{\tau_{k+1}}(\hat{y}_k - \hat{y}_{k-1})]_+$
10: **until** $[r_k]_+ \in \emptyset$

However, often in practice the convergence is not as fast as expected due to oscillations. In [27] the authors discuss a simple yet effective solution first proposed by Chambolle and Dossal: step 8 of Algorithm 9 is replaced by $\tau_{k+1} \leftarrow \frac{k+75}{75}$. This version will be called FISTA-CD.

Whereas with FISTA-BT the factor $\frac{\tau_k - 1}{\tau_{k+1}}$ rises to 0.5 in iteration 4 and to 0.95 in iteration 5, the adjusted factor of FISTA-CD rises to 0.5 in iteration 75, and to 0.95 in iteration 1500. FISTA-CD is thus relaxed version of FISTA-BT. The authors of [27] propose the value 75 in the update of τ, but they comment that there is no proper justification for specifically this value. In general, the higher this value, the later the previous dual vector gets significant weight on the new dual update. The optimal value has to chosen by trial and error.

In Figure 5-4 the two versions of FISTA are simulated on the dual decomposition algorithm. As in FISTA-BT the multiplication factor rises very quickly, the algorithm in combination with $\rho = 3 \times 10^{-5}$ diverges directly. Therefore, a smaller $\rho = 1 \times 10^{-7}$ is chosen: the largest for which the algorithm did not diverge. However, it does not converge either. The other algorithms are simulated with $\rho = 3 \times 10^{-5}$. FISTA-CD clearly accelerates the convergence rate significantly.

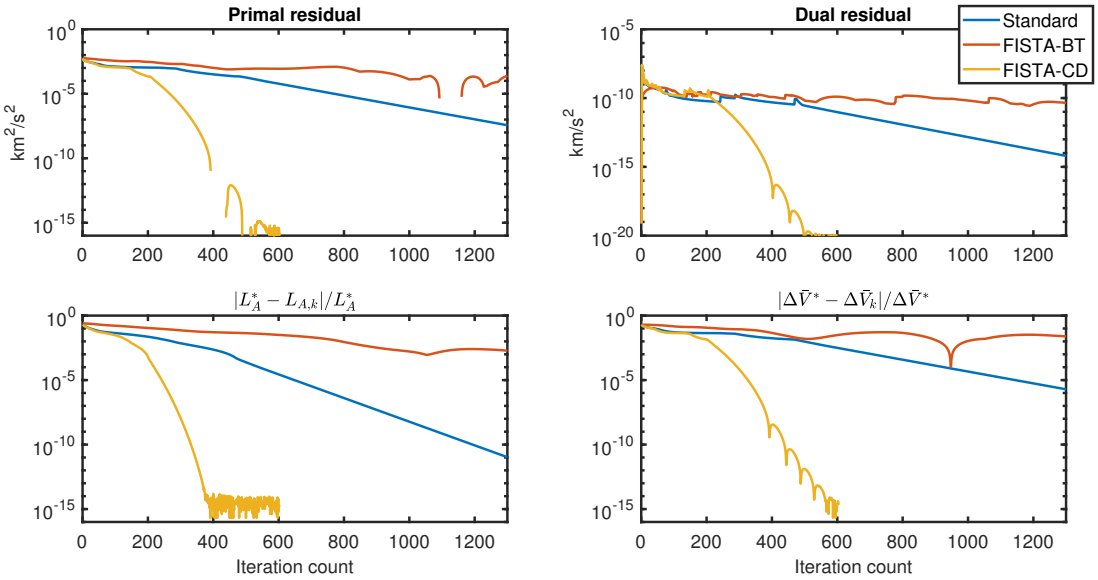


Figure 5-4: The two forms of FISTA in comparison with standard dual decomposition. Standard and FISTA-CD are simulated with $\rho = 3 \times 10^{-5}$; for FISTA-BT the largest converging stepsize was $\rho = 1 \times 10^{-7}$. FISTA-CD can significantly accelerate convergence.

In Figure 5-5 the two forms of FISTA are simulated on synchronous J-ALM. Just like with heavy ball acceleration, U can be updated using a similar FISTA rule as y, or only y can be updated using the FISTA update rule. It is clear that updating both U and y does not converge. It is interesting to see that FISTA-CD first converges, but when multiplication factor $\frac{\tau_k - 1}{\tau_{k+1}}$ becomes too large, the algorithm starts to oscillate and diverge. This is not happening when only U is updated with the standard J-ALM update. In Figure 5-6 the convergence of FISTA-CD is compared with standard J-ALM: interesting enough the convergence rate is increased when a slightly smaller ρ is chosen. The explanation is likely similar to the one discussed in Section 4-2-2.

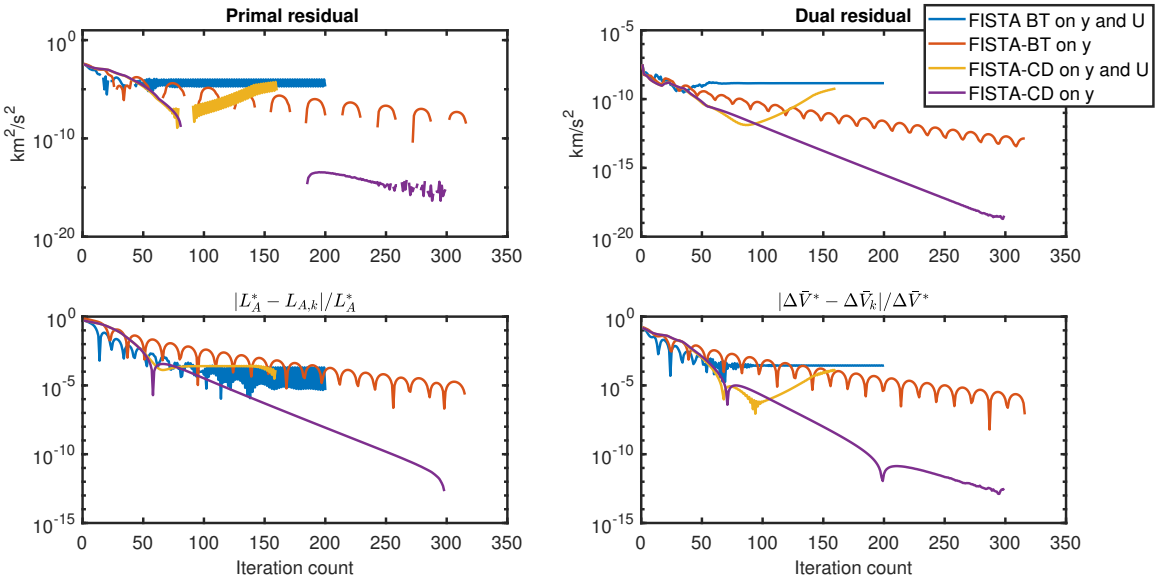


Figure 5-5: Synchronous J-ALM with different implementations of FISTA. All algorithms are simulated with $\rho = 7 \times 10^{-4}$ and $\nu = 0.5$. When U is updated using the same momentum rule as y, the algorithm diverges; only FISTA-CD on y converges.

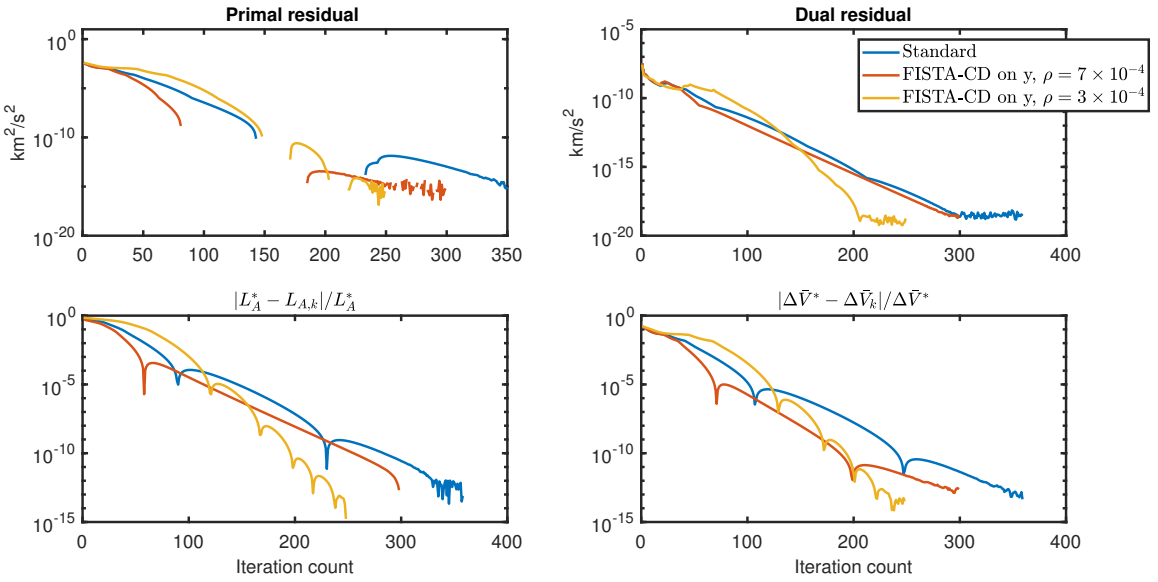


Figure 5-6: Synchronous J-ALM compared to FISTA-CD. When ρ is lowered from $\rho = 7 \times 10^{-4}$ to $\rho = 3 \times 10^{-4}$, the convergence rate increases.

Any form of FISTA on asynchronous J-ALM diverged. This is not unexpected: the multiplication factor in FISTA increases asymptotically to 1, but the heavy ball acceleration simulations showed that the algorithm did not converge already for $\beta = 0.5$.

5-1-3 Anderson Acceleration

This acceleration method searches for the solution of a fixed point iteration problem, which means that it searches for a specific y for which the subgradients are zero. The new vector y is constructed out of a linear combination of previous dual vectors, and the weight vector a is obtained by solving the following minimization problem:

$$a = \underset{a=[a_0,\dots,a_m]^T}{\operatorname{argmin}} \|F_k a\|_2 \quad \text{s.t.} \sum_{i=0}^{m} a_i = 1 \tag{5-4}$$

Here F_k is a matrix containing the subgradients of the dual variable of iteration $k-m+1$ till $k-1$ and $m = \min\{k, m_k\}$ is the number of previous residuals and dual vectors used. The AA update scheme finds the solution of the fixed point iteration problem if (close enough to the optimum) the subgradient of y is linearly related to y. As in AA the new dual variable vector is constructed out of previous dual variable vectors, it can be interpreted as a momentum acceleration method just like HBA and FISTA, but with variable weights and a memory of more than one previous iterate. Therefore AA should be able to obtain even higher convergence rates. Just like the other two acceleration methods, one agent can collect all the control vectors and update the dual vector. However, due to the optimization problem, the AA dual update is significantly more complex than the HBA or FISTA update.

In Figure 5-7 convergence of AA on dual decomposition is shown. The maximum number of previous dual vectors taken into account, m_k has a large effect on the convergence rate: for $m_k = 10$ the algorithm oscillates without showing any sign of converging, but for $m_k = 5$ or 3 the convergence rate is accelerated significantly. These results are obtained using a scaled residual matrix $F_k/\|F_k\|_2$, as the numerical accuracy of the solver influenced the solutions when the algorithm started to converge. The oscillations for $m_k = 10$ can be explained by the fact that the sum of the weight vector a is constrained to be one: the higher the value of m_k, the smaller the weight on the last dual vector.

The simulations of AA on synchronous J-ALM are shown in Figure 5-8. Here $\rho = 4\times10^{-4}$ and $\nu = 0.5$. The algorithm with Anderson update of only the dual vector (AA on y) is compared to Anderson updates of both the dual vector and control acceleration vectors (AA on y and U). Again, updating U in a similar way as y results in the most significant acceleration. Parameter $m_k = 5$ slightly outperforms $m_k = 2$, but this difference is only marginal.

Asynchronous J-ALM did not converge when combined with regular Anderson Acceleration, neither when only y was updated using the Anderson update, nor when both y and U were updated this way. One hypothesis to explain this is the possibility that in the asynchronous algorithm sometimes a dual update is required which temporarily increases the residuals, instead of monotonic decreasing residuals. This might happen as only some of the control vectors are updated per iteration. AA however aims to minimize the residual vector in each iteration, which would prevent the algorithm from finding the optimal solution.

A second hypothesis is that the asynchronous algorithm cannot deal well with large values in a. As only the sum of a is constrained to be 1, the individual values in a may take on large values. If the control vectors with which the update is performed are inaccurate, large values in a may also lead to inaccurate dual vector updates. A simulation of AA with an extra lower bound $a \geq -0.3$ is shown in Figure 5-9. Now AA on asynchronous J-ALM does converge,

it is slightly accelerated when compared to the standard asynchronous J-ALM. It might also be the case that constraining a from below results in solutions with give more weight to the latest dual vector, which in turn eventually results in convergence. The convergence of this accelerated algorithm should be investigated more before more sound conclusions can be drawn.

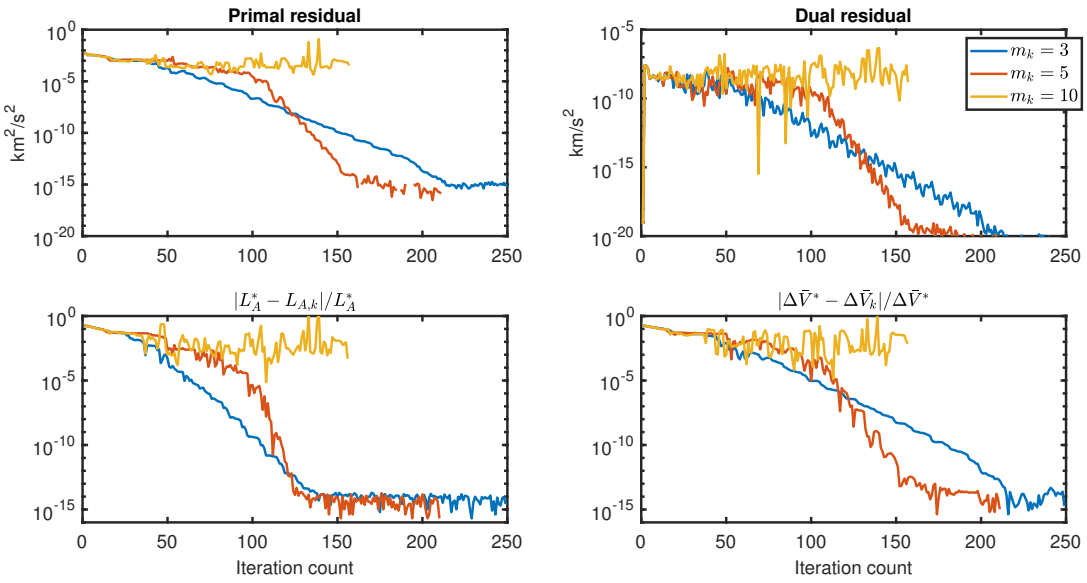


Figure 5-7: Anderson Acceleration on dual decomposition with different values of m_k. All simulations are performed with $\rho = 3 \times 10^{-5}$.

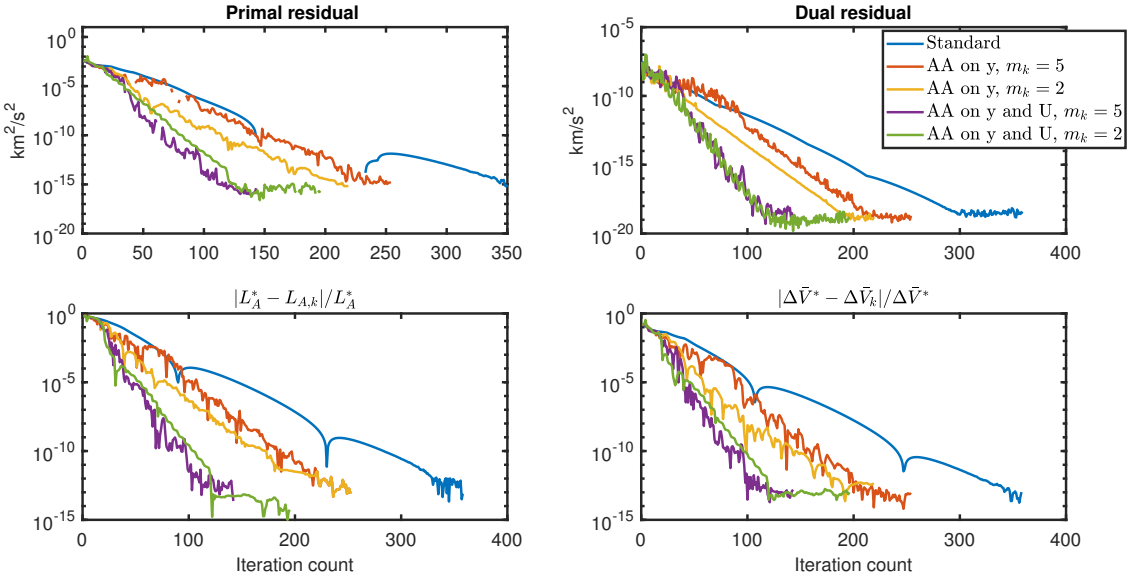


Figure 5-8: Anderson Acceleration on synchronous J-ALM with $\rho = 4 \times 10^{-4}$; again the convergence rate is largest when U is updated using a similar rule as y. For this setup, $m_k = 2$ and $m_k = 5$ perform nearly similar.

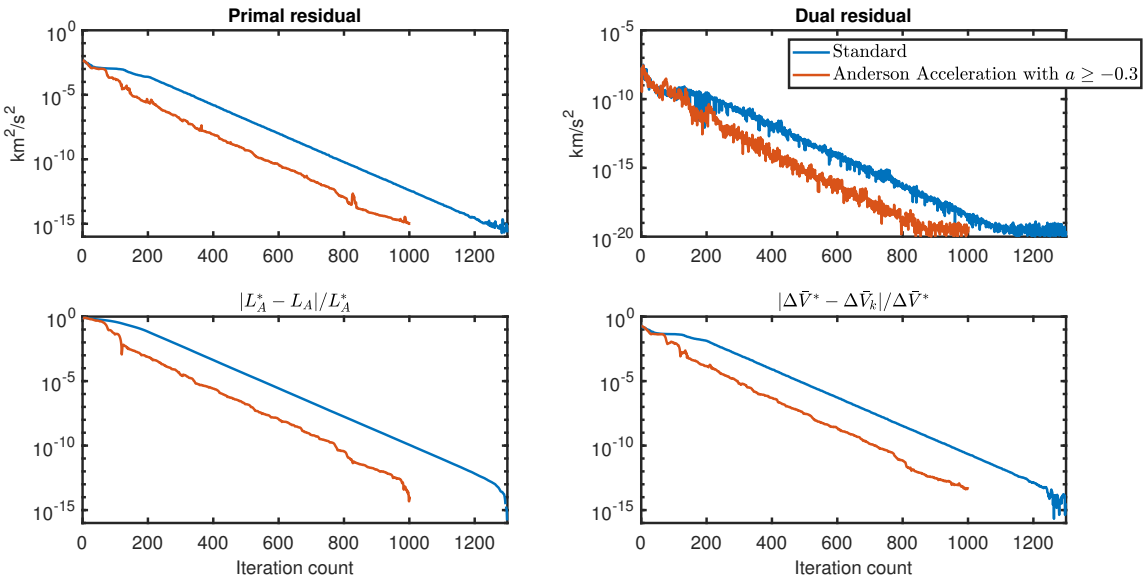


Figure 5-9: Standard versus Anderson Accelerated asynchronous J-ALM with $\rho = 1.4 \times 10^{-4}$; when a lower bound on a is included, Anderson Acceleration can accelerate convergence.

5-1-4 Summarizing Acceleration Figures

To be able to compare the different acceleration methods on the three dual methods, the best results per dual method are given in one figure.

In Figure 5-10 all dual decomposition acceleration algorithms are compared to regular dual decomposition. All simulations are performed with $\rho = 3 \times 10^{-5}$. For HBA, $\beta = 0.9$, and AA is simulated with $m_k = 3$. It is clear that AA outperforms the other acceleration methods; for dual decomposition a varying momentum parameter thus has a large influence on the convergence rate. HBA and FISTA-CD converged in a similar number of iterations; although HBA with its constant parameter β started faster, the convergence rate of FISTA-CD increased significantly over the iterations due to the increasing momentum parameter.

In Figure 5-11 standard synchronous J-ALM is compared to the three acceleration methods. All algorithms are simulated with $\rho = 4 \times 10^{-4}$, which differs from the HBA and FISTA simulations in previous sections as this smaller stepsize resulted in faster convergence. Furthermore, $\nu = 0.5$, HBA is simulated with $\beta = 0.5$ and AA with $m_k = 2$. The resulting figure looks comparable to the dual decomposition overview: AA is by far the fastest, HBA and FISTA reach convergence in the same number of iterations although HBA starts faster, and all three acceleration methods outperform the standard synchronous J-ALM. It is interesting to note that the total weight of each dual vector update is actually higher for HBA and FISTA than for AA. In the latter, the sum of a is constrained to be one, no matter how many previous iterations are taken into account in the update. In HBA, the total weight is given as $1 + \beta \geq 1$ and in FISTA-CD it is $1 + \frac{\tau_k - 1}{\tau_{k+1}} \geq 1$. Therefore it is expected that AA can achieve an even higher convergence rate when the bound on the sum of a is increased.

Finally in Figure 5-12 asynchronous J-ALM is compared to HBA with $\beta = 0.3$ and AA with $m_k = 3$ and $a \geq -0.3$. For all simulations, $\nu = 0.5$, $T_{\max} = 5$ and the dual vector

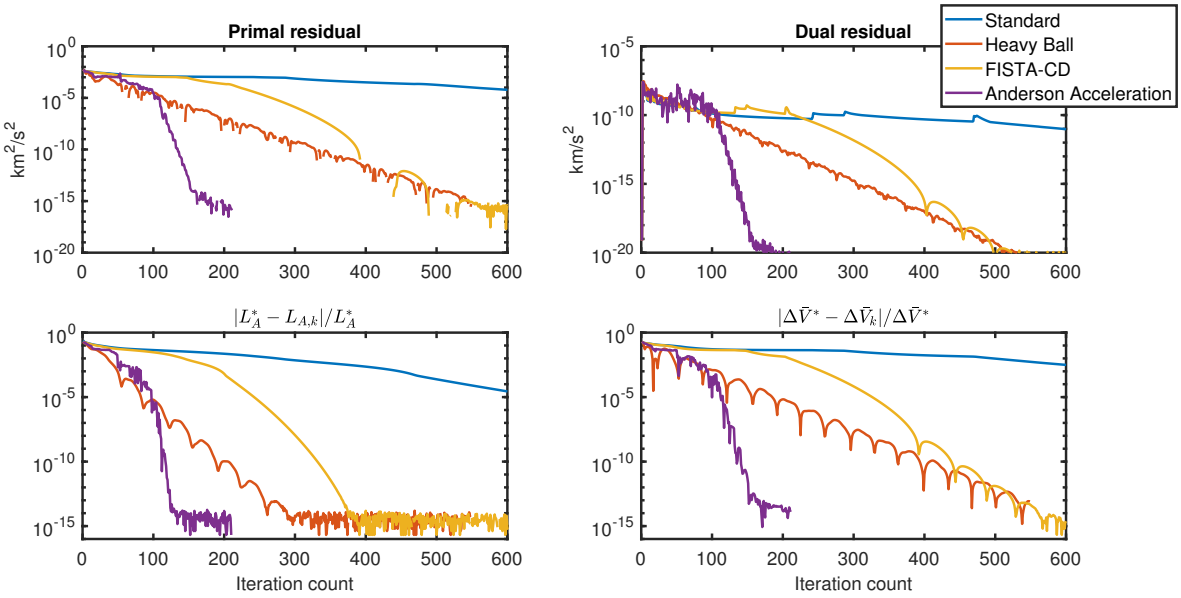


Figure 5-10: Regular dual decomposition compared with different acceleration algorithms. All algorithms are simulated with $\rho = 3 \times 10^{-5}$, Anderson Acceleration clearly outperforms the other acceleration methods.

is updated with two updated control vectors. For both acceleration methods it was difficult to find settings which resulted in convergence; only small acceleration is obtained. As the dual decomposition and synchronous J-ALM simulations showed that AA significantly outperformed other acceleration methods, it is expected that AA on asynchronous J-ALM can also be improved. This requires more insight in the asynchronous convergence characteristics.

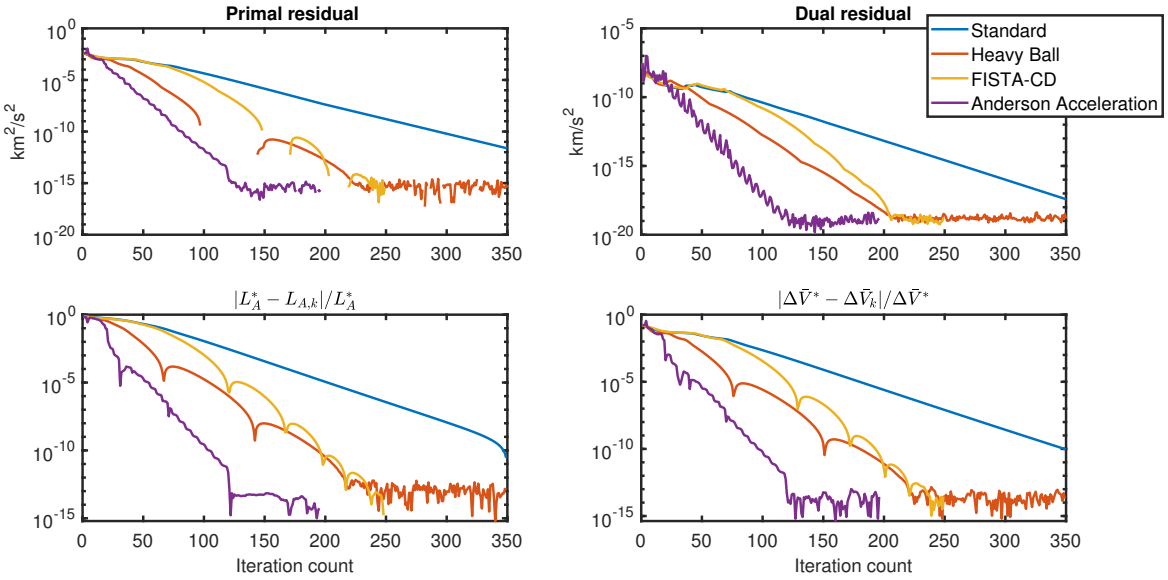


Figure 5-11: Standard synchronous J-ALM compared with three different acceleration algorithms, all with $\rho = 4 \times 10^{-4}$ and $\nu = 0.5$.

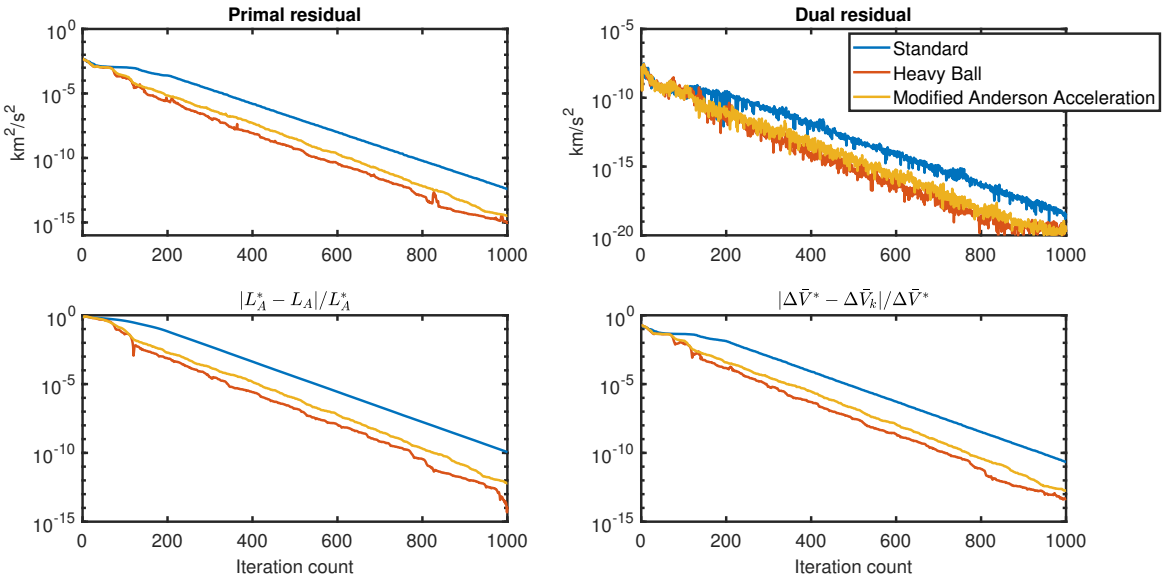


Figure 5-12: Standard asynchronous J-ALM compared with Heavy Ball and Anderson Acceleration, all with $\rho = 1.4 \times 10^{-4}$ and $\nu = 0.5$.

5-2 Simulations with Sequential Convex Programming

In the previous simulations of the dual methods, they are compared on the rate with which they solved only one SCP iteration. In this section Anderson accelerated dual decomposition and J-ALM are simulated with SCP. As stopping criteria, $pr < 1 \times 10^{-6}$ and $dr < 1 \times 10^{-11}$ are chosen. First a four spacecraft reconfiguration is simulated, then the performance of the algorithms on a ten spacecraft reconfiguration is checked using simulations as well to see how the algorithm scales with the number of spacecraft.

5-2-1 Four Spacecraft Reconfiguration

In Figure 5-13 the convergence characteristics of AA on dual decomposition with SCP is shown for $\rho = 5 \times 10^{-3}$ and $m_k = 3$. The SCP algorithm converged in 7 iterations and the solution equals the centralized solution. The SCP iterations are clearly visible in the figure; as the dual vector is initialized as a zero vector the primal residual increases at the start of each SCP iteration. The algorithm converges in a total of 671 dual iterations, the computation time is 624 seconds in total. Despite the fact that in each SCP iteration essentially a new problem is solved due to the update of the collision avoidance constraint, the algorithm is able to converge without changing its parameter values. In Figure 5-14 the simulation of AA on synchronous J-ALM with SCP is shown. This algorithm completes the 7 SCP iterations in 245 dual iterations, in a total of 823 seconds.

In both Figures 5-13 and 5-14 the lower two plots with scaled Lagrangian function value and scaled $\Delta\bar{V}$ seem to remain constant for long periods of time. This has to do with the fact that they both are scaled with L^* and $\Delta\bar{V}^*$ obtained at the end of the last SCP iteration. As the previous SCP iterations converge to different values, the scaled distance to the optimal values remains almost constant.

5-2-2 Ten Spacecraft Reconfiguration

In Figure 5-15 AA on dual decomposition with SCP is applied to a ten spacecraft reconfiguration with $\rho = 3 \times 10^{-5}$ and $m_k = 3$. This specific reconfiguration converged in 4 SCP iterations, but the four iterations differ a lot in terms of convergence rate. This indicates that the parameter selection might be optimized for the different SCP iterations. The algorithm converged in a total of 335 iterations, which is less than the algorithm on the four spacecraft scenario. The required number of iterations thus depends more on the number of collisions which have to be avoided and the proximity of the spacecraft during the reconfiguration than on the absolute number of spacecraft. The total computation time was 628 seconds. In Figure 5-16 the same ten spacecraft reconfiguration scenario is solved using AA on synchronous J-ALM with SCP where $\rho = 1 \times 10^{-4}$ and $m_k = 3$. This algorithm converged in a total of 220 iterations and 3429 seconds.

5-2-3 Computation Times

The convergence characteristics of the SCP simulations are summarized in Table 5-1.

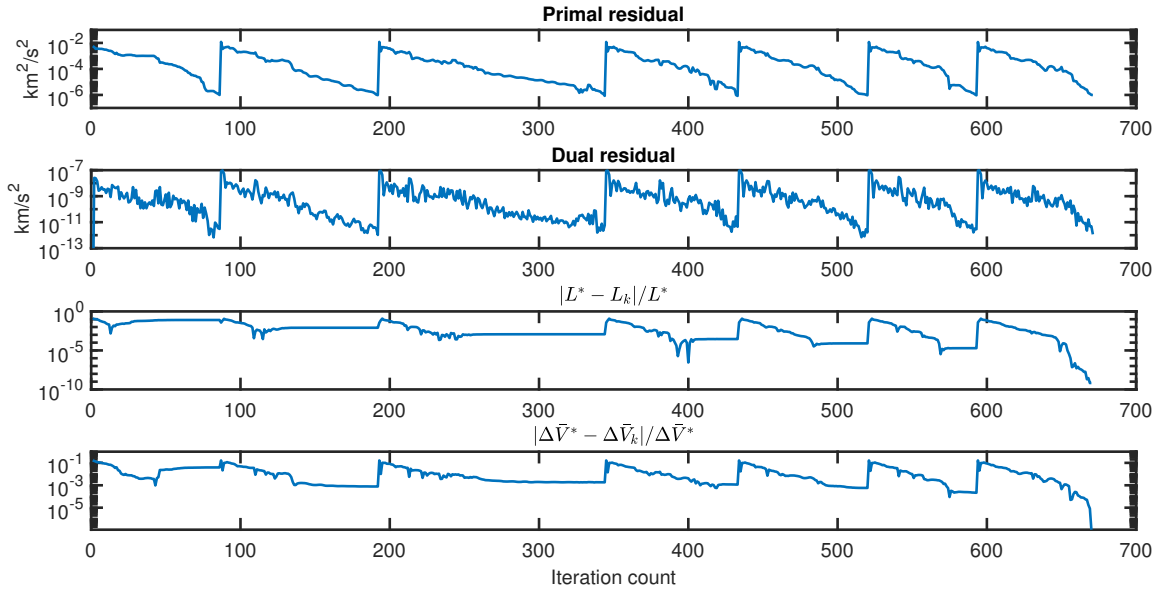


Figure 5-13: Four spacecraft: AA on dual decomposition with SCP. The algorithm converges consistently despite the updated collision avoidance constraint in each SCP iteration. Convergence in 7 SCP iterations.

	N	SCP iter.	Dual iter.	MATLAB time [s]
Dual decomp.	4	7	671	624
Dual decomp.	10	4	335	628
Sync. J-ALM	4	7	245	823
Sync. J-ALM	10	4	220	3429

Table 5-1: Convergence characteristics of Anderson Acceleration on dual algorithms in SCP.

All the dual method with SCP simulations are performed using the MATLAB "parfor" function in combination with 2 workers. To accurately compare the different computation times, the MATLAB computation time has to be divided by $N/2$. The resulting computation time per spacecraft is given in Table 5-2 for dual iterations and SCP iterations. This table also includes the computation times for the hybrid algorithm for comparison.

It is clear that dual decomposition scales better with the number of spacecraft than J-ALM; whereas in dual decomposition the computation time per iteration per spacecraft decreases for the larger formation, in J-ALM the time almost doubles. This is probably caused by the fact that in dual decomposition the collision avoidance constraint is included as affine expressions in the objective, whereas in J-ALM the collision avoidance constraints are included as a separate quadratic expression per constraint. Also the augmented Lagrangian in J-ALM makes use of slack variables which increases the number of optimization variables. The Lagrangian in dual decomposition does not have this, and is thus simpler to minimize. It might be possible to reformulate the objective of the J-ALM minimization problem to a matrix formulation, in order to lower the computation times. Also, all-to-all communication is assumed in this research. A lower communication radius will decrease the problem complexity

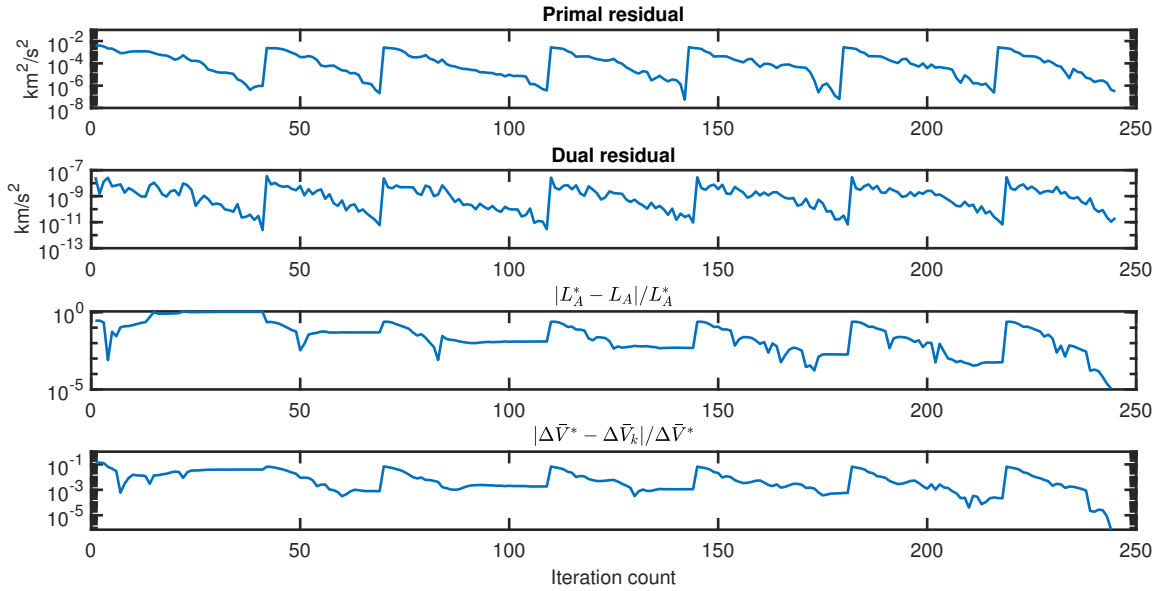


Figure 5-14: Four spacecraft: AA on synchronous J-ALM with SCP. Convergence in 7 SCP iterations.

	N	Time/spacecraft/dual iter. [s]	Time/spacecraft/SCP iter. [s]
Dual decomp.	4	0.47	44.6
	10	0.37	31.4
Sync. J-ALM	4	1.68	58.8
	10	3.12	171.5
Hybrid	4	-	0.5506
	10	-	0.4420

Table 5-2: Time to solve one dual iterations and to solve one SCP iteration for the two accelerated dual methods.

for larger formations. At present both dual methods require two orders of magnitude more time to solve one SCP iteration than the hybrid algorithm, due to the iterative nature of the dual methods.

One disadvantage of obtaining exactly the centralized solution is that the SCP algorithm might not converge, as was discussed in Section 3-4 for the 100 spacecraft scenario. In each SCP iteration all spacecraft trajectories are optimized again. For large formations it might happen that at least one spacecraft will adjust a little each iteration, preventing the convergence tolerance to be met for all spacecraft at the same time. One way to prevent this is to fix the trajectories of spacecraft once they meet the SCP convergence tolerance. This way, both the convergence iteration of SCP and the computation time per iteration can be significantly decreased. How such approximation affects the total required velocity change of the formation could be investigated in future research.

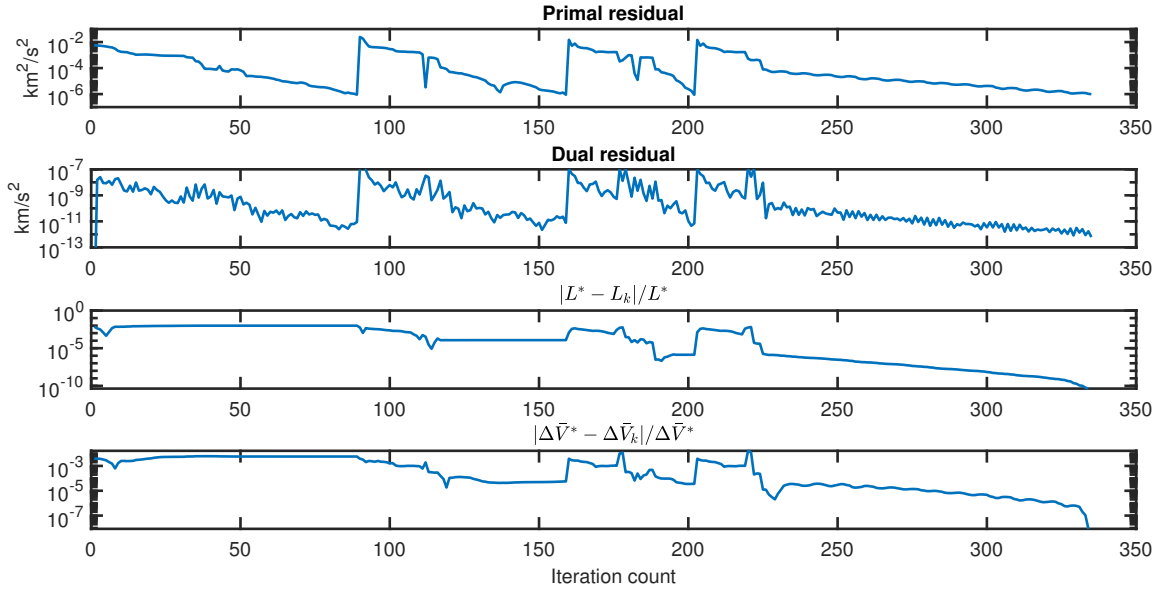


Figure 5-15: Ten spacecraft: AA on dual decomposition with SCP. Convergence in 4 SCP iterations.

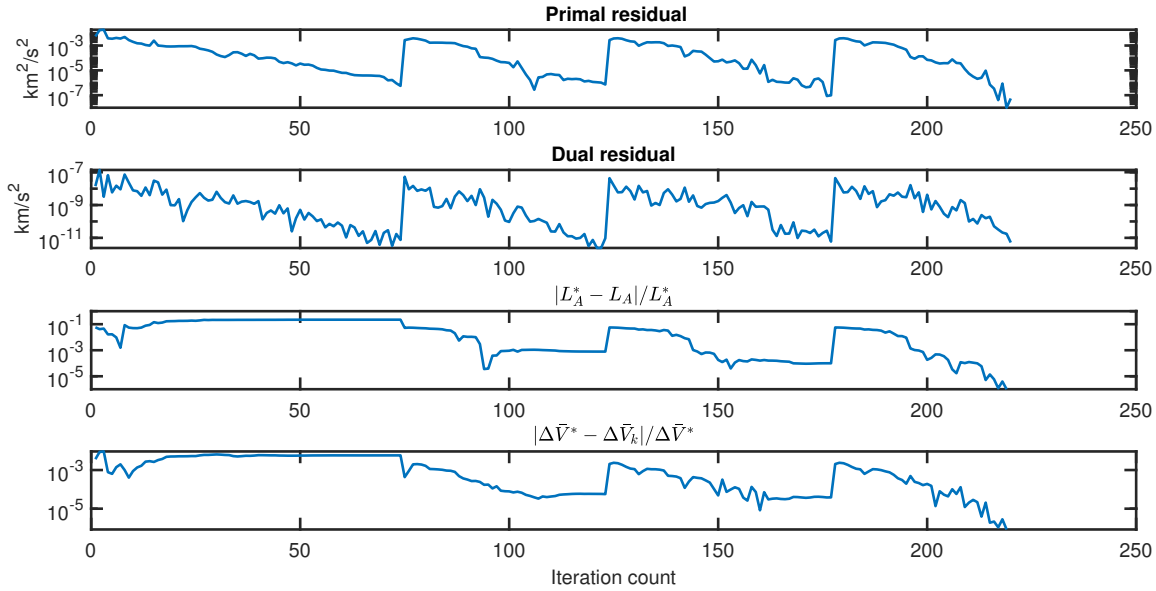


Figure 5-16: Ten spacecraft: AA on synchronous J-ALM with SCP. Convergence in 4 SCP iterations.

5-3 Summary and Conclusions

In this chapter three different acceleration methods have been implemented on dual decomposition and J-ALM, respectively HBA, FISTA and AA. It is shown that on dual decomposition and synchronous J-ALM, AA can significantly accelerate the convergence, outperforming the other two acceleration methods. Acceleration on asynchronous J-ALM acceleration is more difficult as both the dual vector update and the control vector minimization is always performed with inaccurate data. HBA and a modified version of AA were able to accelerate convergence slightly, but more insight is needed to make full use of the acceleration methods.

Dual decomposition and synchronous J-ALM in combination with AA are also simulated with SCP, to compare them to the hybrid cooperation algorithm and to see whether convergence can also be obtained without changing the simulation parameters. It is shown that dual algorithms in SCP do converge, but the varying convergence rates per SCP iteration indicate that likely the simulation parameters can be optimized. Finally it is shown that the algorithms also converge on a reconfiguration of ten spacecraft. For dual decomposition the computation times per spacecraft are similar for a four and a ten spacecraft reconfiguration. For J-ALM the computation time almost doubles however. One way to make the algorithm better scalable is to lower the communication radius: this directly lowers the number of neighbors and thus the problem complexity. It might also be possible to reformulate the objective of the J-ALM minimization problem in order to lower the computation times. This is something which could be researched in future work. For larger formations it is also necessary to adjust the SCP scheme, as the centralized algorithm did not converge on a 100 spacecraft scenario. The dual methods find exactly the same solution as the centralized algorithm, thus will suffer the same drawback. This could be solved by fixing trajectories once they meet the SCP convergence tolerance. How this effects the total required velocity change is not yet known.

Finally, due to the many dual iterations, much data has to be shared between spacecraft in order to obtain collision free reconfiguration trajectories. This might consume more power than is available onboard spacecraft. Therefore at present the hybrid algorithm seems to outperform dual methods on the collision-free trajectory optimization problem.

Chapter 6

Conclusions

In this chapter, first a summary of the research is given with answers to the research sub-questions. Then the main research question is answered, followed by suggestions for future work.

6-1 Summary of the Research and Answers to Sub-Questions

In this research, a distributed collision-free minimum fuel trajectory optimization is developed for the reconfiguration of a large formation of picosatellites.

The relative dynamics of the spacecraft are modeled using the Xu Wang exact dynamical model [37]. This model includes the J2 potential but neglects the atmospheric drag and solar pressure perturbations. These are non-dominant for the simulations performed in this research. Using this linearized and discretized dynamical model and a convex approximation of the collision avoidance constraints, a convex optimization problem is defined. For each pair of spacecraft a separate collision avoidance constraint is introduced.

This convex problem is first solved by a centralized solver. SCP is used to iteratively improve the collision avoidance approximation. Although the solution of this algorithm is not optimal as it excludes large parts of the solution space by approximating the collision avoidance constraints, it provides a good benchmark to compare other algorithms against. In the centralized algorithm both spacecraft for which a collision avoidance constraint is defined are responsible to avoid each other while minimizing the combined cost. The computations of this algorithm can be distributed over all spacecraft in the formation by removing the cooperation in the formation. In that case only one spacecraft per pair is responsible to avoid a collision. This naturally leads to a higher fuel consumption of the total formation. In this research a hybrid algorithm is proposed. Spacecraft on track to collide form cooperating pairs which can solve a small centralized problem. All spacecraft not in a cooperating pair solve the distributed problem.

The sub-questions as given in the introduction are answered as follows.

1: How does the fuel consumption of a collision-free reconfiguration maneuver depend on the degree of cooperation within the formation? The total required velocity change ΔV and thus the required fuel for a collision-free reconfiguration maneuver depends on many factors such as distance the spacecraft have to travel to reach their desired state if they neglect collision avoidance, the number of potential collisions and the amount of cooperation in the formation. A centralized solution with maximum cooperation always finds a better solution than a distributed solution without cooperation if they solve the same problem. Due to the SCP framework, after one iteration the centralized and distributed algorithm actually solve a different problem. Therefore in rare cases it might happen that after convergence of the SCP algorithm the distributed algorithm has found a better solution than the centralized algorithm. In the simulations in this thesis this did not happen however. The hybrid algorithm with cooperating pairs results in a total required velocity change which is close to the centralized solution. Especially for reconfigurations where many collisions have to be avoided, the hybrid algorithm significantly outperforms the distributed algorithm in terms of fuel consumption. For larger formations, the effect of distributed calculations on the computation time per iteration is clear: already for ten spacecraft the distributed and hybrid algorithms are faster than the centralized algorithm.

Instead of decreasing the cooperation in the formation, the coupled convex problem can also be solved in a distributed way using dual methods. Dual decomposition and Consensus ADMM are already proposed in literature to solve the collision-free trajectory optimization problem. In this research the two methods are compared with decomposed Augmented Lagrangian Methods.

2: Would J-ALM be better suited to solve the trajectory optimization problem than dual decomposition and Consensus ADMM? All dual methods discussed in this research are able to find the same solution as the centralized algorithm in finite time. It is shown that synchronous J-ALM indeed outperforms both dual decomposition and Consensus ADMM in terms of convergence rate. Also, calculations in J-ALM are performed in parallel onboard the spacecraft. Finally, in J-ALM the solution can converge almost monotonically to the collision-free solution, whereas Consensus ADMM converges with many oscillations. Therefore it is concluded that J-ALM is better suited to solve the collision-free trajectory optimization problem then dual decomposition and Consensus ADMM.

3: Under what conditions does an asynchronous implementation of a dual method converge? This research demonstrates that J-ALM can also converge in an asynchronous framework, in which the updated control vectors are returned with a maximum relative delay of a specified number of dual iterations. Both the dual update and the control acceleration minimization are thus performed using partly outdated data. It is shown trough simulations that when the stepsize suitable for synchronous J-ALM is divided by the maximum delay, the asynchronous algorithm converges in less iterations than the convergence iteration of synchronous J-ALM multiplied with the maximum delay. When solvers of the spacecraft in a synchronous implementation have large idle times, the asynchronous implementation of J-ALM can thus potentially decrease the absolute convergence time.

4: How do acceleration techniques such as HBA, FISTA and AA influence the convergence rate of both the synchronous and asynchronous dual methods? In this research it is demonstrated that all three acceleration algorithms can significantly accelerate both dual decomposition and synchronous J-ALM. The HBA results in a nearly linear convergence rate on logarithmic scale, whereas FISTA starts slower but gradually increases the convergence rate on the logarithmic scale, reaching convergence at the same time as HBA. AA, originally proposed for fixed-point iteration problems but applied for the first time to ALM in this research, results in nearly linear convergence rate on logarithmic scale as well, while being significantly faster than both HBA and FISTA. On asynchronous J-ALM both HBA and a slightly modified version of AA are shown to be able to accelerate the convergence, although the effect of both acceleration methods is only small.

5: Would dual methods be a feasible alternative to the diminished cooperation algorithms for solving the collision-free trajectory optimization problem in real time? Both dual decomposition and synchronous J-ALM in combination with AA are simulated with SCP for a scenario with four and with ten spacecraft. A solution is obtained in finite time in all simulations. This means that despite the problem changing in each SCP iteration due to the update of the collision avoidance constraint, the parameters do not have to be updated. It is however not yet investigated whether the convergence rate of the algorithms can be increased using adaptive problem parameters. Especially dual decomposition is shown to scale well with the number of spacecraft, although decreasing the computation time for J-ALM in combination with large formations might be possible. Whether dual methods form a feasible alternative to diminished cooperation methods depends on physical spacecraft characteristics such as the power consumption of communications, the reliability of communication links and the communication delay.

6-2 Conclusion of the Research

The main research question was stated as follows:

How to design a distributed collision-free trajectory optimization algorithm for a large formation of spacecraft which finds the minimum-fuel solution in finite time?

Within the constraints and assumptions as stated in the introduction of this thesis, there are two feasible options at this moment. The first is the hybrid algorithm, which is scalable due to parallel computations and flexible as spacecraft only avoid other spacecraft with which they can communicate. This algorithm however results in a solution with slightly more fuel consumption than a centralized solution. The second is the Anderson Accelerated dual decomposition. Although it required more dual iterations than accelerated J-ALM, it scales better to larger formations. It finds the exact same solution as the the centralized algorithm, but compared to the hybrid algorithm it takes much more time to find the solution due to the iterative nature of dual methods. Another drawback of AA on dual decomposition is the selection of the problem parameters. The stepsize ρ and memory parameter m_k have to be selected properly, as they have large influence on the convergence rate. Also different reconfiguration maneuvers require different parameters to obtain the highest convergence rate.

At present the parameters selection is performed by trial and error, which is a time consuming process and not suitable for a real-time trajectory optimization algorithm. AA on J-ALM has the potential to outperform AA on dual decomposition, as it requires approximately only half the iterations. To make it a feasible alternative, the computation time per dual iteration should be decreased for larger formations.

When a real scenario is considered subject to uncertainties, delays, noise and limited available power, it is likely that the dual methods are not feasible to implement. Spacecraft need power to send and receive data, so there is a limit on the feasible amount of communication between spacecraft. In this research it is shown that AA has trouble accelerating asynchronous J-ALM. This problem has to solved at least before communication delays and losses are included. Therefore, at this moment the hybrid algorithm is the preferred choice, although the potential of dual methods to save the extra percentages of fuel is recognized.

When looking at this research with a broader view, it can also be seen as an overview of the possibilities and limitations of various distributed algorithms. Of course the algorithms discussed in this research are not limited to spacecraft reconfigurations. With small adjustments, for example in the relative dynamical model, the same algorithms can be used for other robotic agents such as drones or vessels. Furthermore, dual methods have been used to solve all kinds of problems in a distributed way, in the introduction power systems optimization and large scale image processing were mentioned. The insights from this research might inspire researchers working in other fields as well, to try new problem formulations or acceleration methods.

6-3 Recommended Future Work

This research is performed under many assumptions. These make it possible to compare the various reconfiguration optimization algorithms with each other, but also make the research theoretical instead of directly applicable. Therefore, applying the algorithms discussed in this research to real-life scenarios would likely result in many interesting insights. Most relevant the question exactly how much power is required to communicate between spacecraft and the question how dual methods converge when subject to uncertainties, noise and communication losses. A large part of this research discusses dual methods, for which many questions remain to be answered.

- How to develop adaptive stepsizes for dual methods? Searching the stepsize which results in largest convergence rate is a time-consuming process at the moment. Also, for each different reconfiguration a different stepsize is optimal. Parameter tuning rules or adaptive stepsizes are necessary for the tuning of ADMM and ALM in order to make the algorithms more flexible for solving different problems. The same point applies to the selection of the relaxation parameter for J-ALM, the momentum parameter for HBA or the memory parameter for AA.
- Does J-ALM outperform any Consensus ADMM algortihm for collision-free trajectory optimization? This research demonstrated that synchronous J-ALM outperformed one specific implementation of Consensus ADMM. A more general comparison is required to strengthen the claim that J-ALM outperforms Consensus ADMM to solve this problem.

- How to make the synchronous J-ALM better scalable with the number of spacecraft? The SCP simulations showed that the computation time per spacecraft almost doubled for a ten compared to four spacecraft reconfiguration. This likely has to do with the problem formulation, it might be possible to decrease the computation time by implementing the problem differently.
- How does dual decomposition converge in an asynchronous framework? In this thesis only J-ALM is simulated with asynchronous updates. However, as dual decomposition seems better scalable to larger formations, it would be interesting to also investigate the convergence characteristics of asynchronous dual decomposition.
- Does asynchronous J-ALM always converge when an appropriate stepsize is chosen and the maximum delay is known? Convergence rate characterization and worst-case complexity bounds are not yet available.
- How to accelerate asynchronous J-ALM even more? All three acceleration methods failed to significantly accelerate the convergence of the asynchronous algorithm, which is likely due to the decreased accuracy of the dual and control vector updates. Anderson Acceleration however showed great promise accelerating synchronous dual methods. More insight is needed to find a way to modify this acceleration method to also accelerate the asynchronous dual methods significantly.

Bibliography

[1] P. Sundaramoorthy, "Lecture slides: Micro-Satellite Engineering (AE4-S10)," February 2018. Space Systems Engineering (SSE), Delft University of Technology.

[2] D. J. Morgan, *Guidance and control of swarms of spacecraft*. PhD thesis, University of Illinois at Urbana-Champaign, 2015.

[3] S. Clark, "A chat with Bob Twiggs, father of the CubeSat," 2014 March. Retrieved from https://www.spaceflightnow.com/news/n1403/08cubesats/.

[4] S. Bandyopadhyay, R. Foust, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Review of formation flying and constellation missions using nanosatellites," *Journal of Spacecraft and Rockets*, vol. 53, pp. 567–578, 2016.

[5] E. Kulu, "Nanosats Database," 2019 June. Retrieved from https://www.nanosats.eu.

[6] S. Radu, M. Uludag, S. Speretta, J. Bouwmeester, E. Gill, and N. Chronas Foteinakis, "Delfi-PQ: The first pocketqube of Delft University of Technology," in *Proceedings of 69th International Astronautical Congress: Bremen, Germany [IAC-18-B4.6B. 5]*, International Astronautical Federation, IAF, 2018.

[7] Planet Labs Inc, "Planet Monitoring," 2019. Retrieved from https://www.planet.com/products/monitoring/.

[8] R. J. Luquette, J. Leitner, K. Gendreau, and R. M. Sanner, "Formation Control for the MAXIM Mission," January 2004. Technical Report 20040081408, NASA Goddard Space Flight Center; Greenbelt, MD, United States.

[9] K. G. Carpenter, C. J. Schrijver, R. G. Lyon, L. G. Mundy, R. J. Allen, J. T. Armstrong, W. C. Danchi, M. Karovska, J. Marzouk, L. M. Mazzuca, *et al.*, "The stellar imager (SI) mission concept," in *Future EUV/UV and Visible Space Astrophysics Missions and Instrumentation*, vol. 4854, pp. 293–302, International Society for Optics and Photonics, 2003.

[10] G. Bonin, N. Roth, S. Armitage, B. Risi, and R. Zee, "The CanX-4&5 formation flying mission: A technology pathfinder for nanosatellite constellations," in *Proceedings of the AIAA/USU Conference on Small Satellites, Logan, UT*, 2013.

[11] D. Maessen, *Relative Navigation for Small Spacecraft.* PhD thesis, Delft University of Technology, 2014.

[12] L. F. Peñin, J. Araújo, and N. Ávila, "Design and evaluation of optimal reconfiguration maneuvers for separated space interferometry," *Acta Astronautica*, vol. 57, no. 2-8, pp. 330–340, 2005.

[13] T. Yang, X. Yi, J. Wu, Y. Yu, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Li, and K. Johansson, "A survey of distributed optimization," *Annual Reviews in Control*, vol. 47, pp. 278–305, 2019.

[14] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2941–2962, 2017.

[15] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play ADMM for image restoration: Fixed-point convergence and applications," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 84–98, 2016.

[16] B. He, L. Hou, and X. Yuan, "On full Jacobian decomposition of the augmented Lagrangian method for separable convex programming," *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 2274–2312, 2015.

[17] J. Chu, *Dynamics, Distributed Control and Autonomous Cluster Operations of Fractionated Spacecraft.* PhD thesis, Delft University of Technology, 2015.

[18] L. Ferranti, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Coordination of multiple vessels via distributed nonlinear model predictive control," in *2018 European Control Conference (ECC)*, pp. 2523–2528, IEEE, 2018.

[19] R. Van Parys and G. Pipeleers, "Distributed model predictive formation control with inter-vehicle collision avoidance," in *2017 11th Asian Control Conference (ASCC)*, pp. 2399–2404, IEEE, 2017.

[20] F. Rey, Z. Pan, A. Hauswirth, and J. Lygeros, "Fully decentralized ADMM for coordination and collision avoidance," in *2018 European Control Conference (ECC)*, pp. 825–830, IEEE, 2018.

[21] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[22] B. He, M. Tao, and X. Yuan, "Alternating direction method with Gaussian back substitution for separable convex programming," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 313–340, 2012.

[23] A. Aytekin, H. R. Feyzmahdavian, and M. Johansson, "Analysis and implementation of an asynchronous optimization algorithm for the parameter server," *arXiv preprint arXiv:1610.05507*, 2016.

[24] K. Lee and R. Bhattacharya, "On the convergence analysis of asynchronous distributed quadratic programming via dual decomposition," *arXiv preprint arXiv:1506.05485*, 2015.

[25] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods; lecture notes of EE392o, Autumn Quarter," 2004-2005. Stanford University.

[26] Y. Nesterov, "A method for solving the convex programming problem with convergence rate $O(1/k^2)$," *Dokl. Akad. Nauk SSSR*, vol. 269, no. 3, pp. 543–547, 1983.

[27] J. Liang and C.-B. Schönlieb, "Faster FISTA," *arXiv e-prints arXiv:1807.04005*, July 2018.

[28] V. V. Mai and M. Johansson, "Nonlinear Acceleration of Constrained Optimization Algorithms," in *ICASSP 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4903–4907, IEEE, 2019.

[29] K. Alfriend, S. R. Vadali, P. Gurfil, J. How, and L. Breger, *Spacecraft formation flying: Dynamics, control and navigation*, vol. 2. Elsevier Astrodynamics Series, 2009.

[30] J. Eyer, *A dynamics and control algorithm for low earth orbit precision formation flying satellites*. PhD thesis, University of Toronto, 2009.

[31] P. A. Capó-Lugo and P. M. Bainum, *Orbital Mechanics and Formation Flying: A Digital Control Perspective*, ch. 3, pp. 37–73. Elsevier, 2011.

[32] D. Morgan, S.-J. Chung, L. Blackmore, B. Acikmese, D. Bayard, and F. Y. Hadaegh, "Swarm-keeping strategies for spacecraft under J2 and atmospheric drag perturbations," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 5, pp. 1492–1506, 2012.

[33] J. Sullivan, S. Grimberg, and S. D'Amico, "Comprehensive survey and assessment of spacecraft relative motion dynamics models," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 8, pp. 1837–1859, 2017.

[34] D. Wang, B. Wu, and E. K. Poh, *Satellite Formation Flying: Relative Dynamics, Formation Design, Fuel Optimal Maneuvers and Formation Maintenance*, vol. 87. Springer, 2016.

[35] G. Gaias, J.-S. Ardaens, and O. Montenbruck, "Model of J2 perturbed satellite relative motion with time-varying differential drag," *Celestial Mechanics and Dynamical Astronomy*, vol. 123, no. 4, pp. 411–433, 2015.

[36] A. W. Koenig, T. Guffanti, and S. D'Amico, "New state transition matrices for spacecraft relative motion in perturbed orbits," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 7, pp. 1749–1768, 2017.

[37] G. Xu and D. Wang, "Nonlinear dynamic equations of satellite relative motion around an oblate Earth," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 5, pp. 1521–1524, 2008.

[38] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2018. Retrieved from http://www.gurobi.com.

[39] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual: Choosing the right algorithm," 2018. Retrieved from https://www.gurobi.com/documentation/8.1/refman/numerics_choosing_the_righ.html.

[40] S. Boyd, L. Xiao, A. Mutapcic, and J. Mattingley, "Notes on decomposition methods; lecture notes of EE364B, Winter Quarter," 2006-2007. Stanford University.

[41] A. Nedić and A. Ozdaglar, "Approximate primal solutions and rate analysis for dual subgradient methods," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1757–1780, 2009.

[42] H. Y. Ong and J. C. Gerdes, "Cooperative collision avoidance via proximal message passing," in *2015 American Control Conference (ACC)*, pp. 4124–4130, IEEE, 2015.

[43] Z. Wang, Y. Zheng, S. E. Li, K. You, and K. Li, "Parallel Optimal Control for Cooperative Automation of Large-scale Connected Vehicles via ADMM," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1633–1639, IEEE, 2018.

[44] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on networking*, no. 3, pp. 375–385, 1996.

[45] C. Chen, B. He, Y. Ye, and X. Yuan, "The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent," *Mathematical Programming*, vol. 155, no. 1-2, pp. 57–79, 2016.

Glossary

List of Acronyms

AA	Anderson Acceleration
ADMM	Alternating Direction Method of Multipliers
ALM	Augmented Lagrangian Method
CanX-4,5	Canadian Advanced Nanospace eXperiment-4&5
Delfi-PQ	Delfi-PocketQube
DOF	Degree of Freedom
ECI	Earth Centered Initial
FISTA	Fast Iterative Shrinkage-Threshold Algorithm
GS-ALM	Gauss-Seidel decomposition of the Augmented Lagrangian Method
HBA	Heavy Ball Acceleration
J-ALM	Jacobian decomposition of the Augmented Lagrangian Method
LEO	Low Earth Orbit
LVLH	Local-Vertical-Local-Horizontal
MPC	Model Predictive Control
PPRO	Passive Periodic Relative Orbits
RAAN	Right Ascension of Ascending Node
ROE	Relative Orbital Elements
RSV	Reference Satellite Variables
SCP	Sequential Convex Programming

STM State-Transition Matrix

SWIFT Silicon Wafer Integrated Femtosatellites

List of Symbols

β	Heavy Ball momentum parameter
κ	Discrete time instance
ϵ_{SCP}	Convergence tolerance for the SCP algorithm
γ	Geocentric latitude
μ	Gravitational parameter of Earth
Ω	Right ascension of the ascending node
π_i	Priority value of spacecraft i
ρ	Stepsize for the dual update
σ	Standard deviation of positions in initial and final configuration
θ	True anomaly
$\bar{O}$	Last block row of O
$\bar{r}$	Position vector of the formation center with respect to Earth
$\bar{T}$	Last block row of T
$\bar{U}_j$	Best available prediction of the control acceleration vector U_j
$\bar{x}$	Best available prediction of the state of a spacecraft
$\hat{G}$	Block diagonal matrix consisting of copies of G
$\hat{r}$	Magnitude of position vector of formation center
$\hat{U}_i^l$	Control acceleration vector of spacecraft i at SCP iteration l
$\hat{X}$	X-vector in ECI frame
$\hat{x}$	X-vector in LVLH frame
$\hat{Y}$	Y-vector in ECI frame
$\hat{y}$	Y-vector in LVLH frame
$\hat{Z}$	Z-vector in ECI frame
$\hat{z}$	Z-vector in LVLH frame
a	Weight vector in Anderson Acceleration
B	Stacked matrices B_{ij}
c_{J2}	Constant to simplify notation of the Xu-Wang dynamics model
D	Lipschitz constant
D_{ij}	Vector which contains the 2-norm of the position differences of predicted trajectories $\bar{X}_i$ and $\bar{X}_j$
F	Matrix used to shorten the notation of collision avoidance constraint residuals for all spacecraft pairs
F_k	Residual matrix for Anderson Acceleration

G	Matrix selecting positions from a state vector
h	Angular momentum vector
i_c	Inclination
J_2	Second zonal harmonic coefficient of the Earth
K	Kinetic energy
k	Iteration number in dual algorithms
L	Lagrangian
l	SCP iteration number
m	Strong convexity parameter
m_k	Maximum number of previous dual vectors taken into account in Anderson Acceleration
M_{ij}	Block diagonal matrix consisting of position differences of predicted trajectories $\bar{X}_i$ and $\bar{X}_j$
N	Number of spacecraft
O	Block matrix with products of matrices A used in dynamical constraint
P	Potential energy
R_{col}	Minimum separation distance between two spacecraft
R_{comm}	Maximum distance for which two spacecraft can still communicate
R_e	Earth's equatorial radius
r_{ij}	Residual vector of spacecraft i and j: a measure for the collision avoidance constraint violation
s	Slack variable vector
T	Block matrix with products of matrices A and B used in dynamical constraint
T	Delay in asynchronous algorithm
$T_{\max}$	Maximum delay in asynchronous J-ALM
T_f	Time at which the spacecraft should have acquired the new formation
$u_{\max}$	Maximum control acceleration per DOF
U_j	Vector containing the control acceleration inputs over entire horizon
$u_{i,t}$	Control acceleration vector in 3 DOF for spacecraft i at time t
v_x	Radial velocity
W_i	Consensus variable vector related to U_i
X_j	Vector containing all states of spacecraft j over the prediction horizon
$x_{i,\text{fin}}$	Desired final state for spacecraft i
$x_{i,\text{in}}$	Given initial state of spacecraft i
$x_{i,t}$	State vector of spacecraft i at time t, consisting of position and velocity in three directions
y_{ij}	Dual vector for spacecraft pair i, j
*	Optimal solution
$[.]_+$	Projection operator to $\mathbb{R}_+$
$\Delta\bar{V}$	Required velocity change to perform the a reconfiguration maneuver, averaged over the number of spacecraft

$\mathbb{R}^d$	Domain of the trajectory optimization problem
$\mathbf{0}$	Zero vector with appropriate size
$\mathcal{C}_i$	Set of neighbors which potentially collide with spacecraft i.
$\mathcal{N}_i$	Set of spacecraft within communication distance R_{comm} of spacecraft i
$\mathcal{P}_i$	Set of spacecraft that spacecraft i has to avoid
$\mathcal{U}_i$	Convex set of vectors U_i which satisfy the dynamical and maximum control acceleration constraints.