Distributed Model Predictive Controller Design Based on Distributed Optimization

Doãn Minh Đăng

Ph.D. Thesis

DISTRIBUTED MODEL PREDICTIVE CONTROLLER DESIGN BASED ON DISTRIBUTED OPTIMIZATION

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op

woensdag 21 november 2012 om 12:30 uur

door

Minh Dang DOAN

Master of Science in Systems and Control
Delft University of Technology
geboren te Cantho, Vietnam

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. ir. B. De Schutter

Copromotor: Dr. ir. T. Keviczky

Samenstelling promotiecommisie:

Rector Magnificus,	voorzitter
Prof. dr. ir. B. De Schutter,	Technische Universiteit Delft, promotor
Dr. ir. T. Keviczky,	Technische Universiteit Delft, copromotor
Prof. dr. ir. Hans Hellendoorn,	Technische Universiteit Delft
Prof. dr. ir. Fred van Keulen,	Technische Universiteit Delft
Prof. dr. ir. Riccardo Scattolini,	Politecnico di Milano
Prof. dr. ir. Anders Rantzer,	Lunds Universitet
Prof. dr. ir. Maurice Heemels,	Technische Universiteit Eindhoven

The work presented in this thesis has been supported by the European Union Seventh Framework STREP project *Hierarchical and Distributed Model Predictive Control (HD-MPC)* with contract number INFSO-ICT-223854.

ISBN: 978-94-6203-214-9

Printed by Wöhrmann Print Service in The Netherlands.

dedicated to my parents

Kính tặng Bố Mẹ

Acknowledgments

Writing this Ph.D. thesis is a continuing process which began in November 2008. Throughout these four years, from single papers to this thesis, a variety of people have left their footprints in my life and contributed to my Ph.D. research in their own ways.

First and foremost, I would like to express my deepest gratitude to my supervisors Bart De Schutter and Tamás Keviczky. In the last four years, they have constantly helped me with writing my research papers as well as this Ph.D. thesis. They have patiently guided me to develop my research skills through their supervision, support and discussions which have been valuable and helpful. Without them, I have never been able to come to this point of my thesis.

Next, I would like to thank Amol, Lakshmi, Samira, Alfredo, Yashar, Bart, Ismini, Nicolas, Max, and Juan for the relaxing chit-chat during the time we worked in the same office. My special thank goes to Amol, who always shared the office with me when we had the overlapping time working in the TU Delft. He brought back uncountable interesting facts and thoughts to our daily conversations, which help provoke my thinking.

Huge thanks to Pontus, Andrea, Felipe, Quốc, Ion, Valentin, and Brett for our discussions and collaborations during my research.

I would like to thank Justin, Paolo, Andrea, Alfredo, Amol, Ali, and Quốc for being my travel companions and sharing pleasant moments when we were attending conferences together.

I also would like to thank my colleagues involving in the HD-MPC project for the collaborations and scientific discussions, and would like especially thank Rudy, Alfredo, Holger, Felipe, Jairo, Laura, Marcello, Attila, and Carlo for our interactions during the meetings and workshops of the project.

I am grateful to the members of my Ph.D. committee for providing me with constructive remarks, which helped me to improve this thesis.

Many thanks to Noortje for helping me with Dutch translation, and Arne and Gijs for proofreading the propositions and summary of my Ph.D. thesis.

I would like to thank my friends in the Vietnamese Community in Delft (VCiD) for the friendship and mutual support we had when we were studying far away from our homeland.

I would like to thank my colleagues in Delft Center for Systems and Control, especially Ph.D. fellows, for the friendly environment with scientific discussions we shared when I

worked in the block 8C of 3mE building. I will remember the lunches, sport activities, coffee breaks, movie nights, and amusing email discussions we have had together.

I would like to thank the Department of Automatic Control at Lund University for hosting me from April to June 2011, with a warm and peaceful atmosphere, as well as active research activities from which I also benefitted.

My thanks also go to my friends and colleagues in Vietnam for their encouragement and understanding when I study abroad. I am grateful to my formers professors at Ho Chi Minh City of Technology for equipping me with a competent scientific background. I am thankful to the Mekong1000 Project and Cantho150 Program for generously funding my M.Sc. study that provided me an opportunity for pursuing this Ph.D study.

Then, I would like to extend my heartfelt indebtedness to my parents for devotedly bringing me up, taking the best care of me and constantly encouraging me to achieve more. Their endless love is the reason I have been trying my best to complete my Ph.D. research, and this thesis is considered as a gift I especially dedicated to them when they have just entered their retirement.

Finally, I am grateful to my wife Cẩm Tâm and my daughter Thủy An for sharing ups and downs during my Ph.D. candidacy in TU Delft. They have been making my days bright despites the Dutch weather, especially my dear daughter, who is growing up and cheering up every day.

Delft, October 2012
Dang

Summary

Due to the size and structure of control design problems for large-scale complex systems, standard approaches using a centralized controller are impractical, and in some cases intractable due to the number and complexity of interactions between subsystems. This fact has stimulated renewed interest and research on distributed control for large-scale complex systems.

This thesis aims to provide tools for designing distributed model predictive controllers for linear systems. The essence of model predictive control (MPC) is to formulate the control problem as a repeated solution to a finite-time optimization problem. This enables straightforward controller design for multi-input multi-output systems with hard constraints, thus making this method widely adopted in the industry. However, for large-scale complex systems, this practice gives rise to several serious issues: a global communication mechanism is needed for sending measurement data to a central node; the complexity of the resulting centralized optimization problem leads to a high computational burden; the whole system's performance and safety depend on the result generated by the single central controller, thus lowering the resilience of the system. In order to design distributed model predictive controllers, our research focuses on solving the resulting large-scale MPC optimization problem in a distributed way.

In particular, this thesis addresses two issues:

1. Designing distributed optimization algorithms for solving convex optimization problems arising in MPC for discrete-time linear systems.

2. Determining conditions for achieving feasibility and stability of the closed-loop system.

The control setting consists of a group of local controllers associated with subsystems that have limited communications among them, and where each controller has a processor for handling local computation tasks. The original centralized MPC optimization problem is formulated as a quadratic program, with a separable cost function and linear constraints, and each constraint involves a small number of subsystems, i.e., there is a sparse coupling pattern introduced by the constraints. In order to solve such problem in a distributed fashion, dual decomposition techniques are used. With a proper definition of the local variables (including states and control inputs) and the subsystem neighborhoods (i.e., the subsystems that can directly interact and communicate with the given subsystem), algorithms using first-order derivatives can be used to solve the dual problem in a distributed

way. We propose three main distributed and hierarchical algorithms: distributed Han's, distributed accelerated proximal gradient, and hierarchical primal feasible using dual gradient algorithms.

First, Fenchel duality is used to formulate the dual function, and the indicator function is used to relax the constraints. The underlying algorithm is Han's parallel method, which belongs to the class of projected gradient algorithms. We show that the main subproblems of Han's method have analytical solutions, and thus the algorithm involves only iterative linear algebra computations, which are cheap and can be implemented in a distributed setting. The resulting distributed Han's method is proved to generate results that are equivalent to those of the centralized counterpart, and thus to converge to the centralized MPC solution at every sampling step. Based on the convergence of the algorithm, feasibility and stability of the closed-loop system are inherited from the original centralized MPC setting.

Next, an accelerated proximal gradient algorithm is used to solve the dual problem that can be obtained by either Fenchel or Lagrange duality. This algorithm belongs to the class of accelerated gradient-based algorithms, which are known to achieve the best convergence rate among all gradient-based algorithms. We show that this accelerated proximal gradient algorithm can be considered as an extended and improved version of Han's algorithm, as it converges one order of magnitude faster than the classical proximal gradient algorithm, which is equivalent to Han's method for quadratic programs. Moreover, by using the indicator function, we can treat a problem with a mixed 2-norm and 1-norm cost function by constructing a differentiable dual function for the nondifferentiable original problem. As the additional computation task for acceleration is only a linear combination of solutions obtained in the two preceding iterations, this accelerated algorithm only needs more memory to store previous iterates, while performing computations that are just as cheap as those of the classical version. Hence, this algorithm can be implemented in a distributed fashion similarly to the distributed Han's algorithm.

In the third method, a two-layer iterative hierarchical approach is used to solve the Lagrange's dual problem of the centralized MPC convex optimization problem. In the outer loop, the dual function is maximized using a projected gradient method in combination with an averaging scheme that provides bounds for the feasibility violation and the suboptimality of the primal function. In the inner loop, a hierarchical optimization algorithm is used to provide either an exact or an approximate solution with a desired precision to the minimization of the Lagrangian function. We present two algorithms for the inner loop: a hierarchical conjugate gradient method and a distributed Jacobi optimization algorithm. This method can be applied to MPC problems that are feasible in the first sampling step and when the Slater condition holds (i.e., there exists a solution that strictly satisfies the inequality constraints). Using this method, the controller can generate feasible solutions of the MPC problem even when the dual solution does not reach optimality, and closed-loop stability is also achieved.

In addition to developing novel algorithms, this thesis also emphasizes implementation issues by considering an application of hydro power production control. We consider the control problem of a hydro power valley with nonlinear system dynamics. Different topics have been considered, including model reduction and reformulation of the MPC optimization problem so that the resulting optimization problem is suitable for applying the distributed algorithms developed in this thesis. We show that by implementing our proposed distributed accelerated proximal gradient algorithm, the distributed controller yields

a performance that is as good as that of a centralized controller, while the distributed algorithm uses remarkably less CPU time for computation than a centralized solver. The results from this application example confirm and support the applicability of distributed MPC on large-scale complex systems.

Samenvatting

Vanwege de omvang en de structuur van regelaarsontwerpproblemen voor grootschalige complexe systemen zijn standaard benaderingen die gebruik maken van gecentraliseerde regelaars onuitvoerbaar, en in sommige gevallen onhandelbaar vanwege het aantal en de complexiteit van de interacties tussen de deelsystemen. Dit feit heeft hernieuwde belangstelling voor en onderzoek naar gedistribueerde regeling van grootschalige complexe systemen gestimuleerd.

Deze dissertatie beoogt hulpmiddelen te ontwikkelen voor het ontwerpen van gedistribueerde model-gebaseerde voorspellende regelaars voor lineaire systemen. De essentie van model-gebaseerde voorspellende regeling (MPC) is het formuleren van het regelprobleem als het herhaald oplossen van een eindige-tijd optimalisatieprobleem. Dit maakt op een eenvoudige manier een regelingsontwerp mogelijk voor systemen met verscheidene ingangen en uitgangen en met harde beperkingen, waardoor deze methode breed wordt toegepast in de praktijk. Voor grootschalige complexe systemen zorgt deze aanpak echter voor een aantal belangrijke kwesties: een globaal communicatiemechanisme is nodig om gegevens naar een centraal knooppunt te versturen; de complexiteit van het resulterende gecentraliseerde optimalisatieprobleem leidt tot een grote rekenlast; de prestatie en de veiligheid van het gehele systeem hangt af van het resultaat dat wordt gegenereerd door de enkelvoudige centrale regelaar, hetgeen de veerkracht van het systeem verlaagt. Met het oog op het ontwerpen van gedistribueerde model-gebaseerde voorspellende regelaars is ons onderzoek gericht op het oplossen van de resulterende grootschalige MPC-optimalisatieproblemen op een gedistribueerde manier.

In het bijzonder richt deze dissertatie zich op twee kwesties:

1. Het ontwerpen van gedistribueerde optimalisatiealgoritmen voor het oplossen van convexe optimalisatieproblemen die optreden bij MPC voor discrete-tijd lineaire systemen.

2. Het bepalen van condities voor het behalen van haalbaarheid (In het Engels: feasibility) en stabiliteit van het gesloten-lussysteem.

De regelsituatie bestaat uit een groep van lokale regelaars die zijn geassocieerd met deelsystemen die beperkte communicatiemogelijkheden met elkaar hebben en waarin elke regelaar een processor heeft om lokale berekeningstaken uit te voeren. Het oorspronkelijke gecentraliseerde MPC-optimalisatieprobleem wordt geformuleerd als een kwadratisch optimalisatieprobleem met een scheidbare kostfunctie en met lineaire beperkingen, waarin

elke beperking een klein aantal deelsystemen beslaat; met andere woorden, er wordt een ijl koppelingspatroon geïntroduceerd door de beperkingen. Teneinde een dergelijk probleem op een gedistribueerde manier op te lossen, worden duale decompositietechnieken gebruikt. Met een adequate definitie van de lokale variabelen (inclusief toestanden en regelingangen) en van de omgevingen van de deelsystemen (namelijk, die deelsystemen die direct kunnen interageren en kunnen communiceren met het gegeven deelsysteem), kunnen algoritmen die eerste-orde afgeleiden gebruiken, worden toegepast om het duale probleem op een gedistribueerde manier op te lossen. We stellen drie gedistribueerde en hiërarchische algoritmen voor: een gedistribueerde methode van Han, een gedistribueerde versnelde proximale-gradiëntmethode, en een hiërarchische primaal-haalbare methode gebruikmakend van duale gradiëntalgoritmen.

Bij de eerste methode wordt Fenchel dualiteit gebruikt om de duale functie te formuleren, en de indicatorfunctie wordt gebruikt om de beperkingen te verzachten. Het onderliggende algoritme is de parallelle methode van Han, dat behoort tot de klasse van geprojecteerde-gradiëntalgoritmen. We tonen aan dat het hoofdzakelijke deelprobleem van de methode van Han analytische oplossingen heeft, en dus gebruikt het algoritme slechts iteratieve berekeningen uit de lineaire algebra, die goedkoop zijn en die geïmplementeerd kunnen worden in een gedistribueerde omgeving. We laten zien dat de resulterende gedistribueerde methode van Han resultaten genereert die equivalent zijn met die van de gecentraliseerde tegenhanger. Bijgevolg convergeert de gecentraliseerde MPC oplossing op elke tijdsstap. Gebaseerd op de convergentie van het algoritme worden haalbaarheid en stabiliteit van het gesloten-lussysteem overgeërfd van de originele gecentraliseerde MPC situatie.

Vervolgens wordt een versneld proximale-gradiëntalgoritme gebruikt om het duale probleem op te lossen dat verkregen kan worden uit Fenchel- dan wel uit Lagrange-dualiteit. Dit algoritme behoort tot de klasse van versnelde gradiënt-gebaseerde algoritmen, welke bekend staan voor het behalen van de beste convergentiesnelheid onder alle op de gradiënt gebaseerde algoritmen. We laten zien dat dit versnelde proximale-gradiëntalgoritme beschouwd kan worden als een uitgebreidere en verbeterde versie van het algoritme van Han, gezien het één orde van grootte sneller convergeert dan het klassieke proximale-gradiëntalgoritme, dat equivalent is aan de methode van Han voor kwadratische optimalisatieproblemen. Bovendien kunnen we een probleem met een gemengde 2-norm en 1-norm kostfunctie behandelen door de indicatorfunctie te gebruiken, en door een differentieerbare duale functie op te stellen voor het originele niet-differentieerbare probleem. Gezien de bijkomende berekeningstaak voor de versnelling slechts een lineaire combinatie inhoudt van oplossingen die verkregen zijn in de twee voorafgaande iteraties, heeft dit versnelde algoritme alleen meer geheugen nodig om voorafgaande iteraties op te slaan, terwijl het berekeningen uitvoert die even goedkoop zijn als die van de klassieke versie. Derhalve kan dit algoritme worden geïmplementeerd op een gedistribueerde manier overeenkomstig met het gedistribueerde algoritme van Han.

In de derde methode wordt een twee-laagse iteratieve hiërarchische aanpak genomen om het dualiteitsprobleem van Lagrange op te lossen van het gecentraliseerde convexe MPC-optimalisatieprobleem. In de buitenste lus wordt de duale functie gemaximaliseerd, gebruikmakend van een geprojecteerde-gradiëntmethode in combinatie met een middelingsschema dat grenzen levert voor de haalbaarheidsschending (In het Engels: feasibility violation) en de suboptimaliteit van de primale functie. In de binnenste lus wordt een hiërarchisch optimalisatiealgoritme gebruikt om ofwel een exacte dan wel een benaderende oploss-

ing te leveren met de gewenste precisie voor de minimalisatie van de Lagrange functie. We stellen twee algoritmen voor de binnenste lus voor: een hiërarchische toegevoegde-gradiëntmethode en een gedistribueerd Jacobi optimalisatiealgoritme. Deze methode kan worden toegepast in MPC problemen die haalbaar zijn op de eerste tijdsstap en wanneer de Slater conditie geldt, dat wil zeggen, dat er een oplossing bestaat die strikt voldoet aan de ongelijkheidsbeperkingen. Door deze methode te gebruiken, kan de regelaar haalbare oplossingen van het MPC probleem genereren zelfs wanneer de duale oplossing geen optimaliteit bereikt en tevens wordt gesloten-lusstabiliteit behaald.

Naast het ontwikkelen van vernieuwende algoritmen benadrukt deze dissertatie ook implementatiekwesties door een toepassing te beschouwen van de regeling van waterkrachtproductie. We beschouwen het regelprobleem van een waterkrachtvallei met niet-lineaire systeemdynamica. Verschillende onderwerpen worden in overweging genomen, inclusief modelreductie en herformulering van het MPC optimalisatieprobleem zodat het resulterende optimalisatieprobleem geschikt is voor de toepassing van de gedistribueerde algoritmen die in deze dissertatie ontwikkeld zijn. We laten zien dat door het implementeren van het door ons voorgestelde gedistribueerde versnelde proximale-gradiëntalgoritme, de gedistribueerde regelaar een prestatie bereikt die even goed is als die van een gecentraliseerde regelaar, terwijl het gedistribueerde algoritme aanzienlijk minder rekentijd gebruikt voor de berekeningen dan een gecentraliseerde aanpak. De resultaten van dit toepassingsvoorbeeld bevestigen en ondersteunen de toepasbaarheid van gedistribueerde modelgebaseerde voorspellende regeling (MPC) voor grootschalige complexe systemen.

Tóm tắt

Do yêu cầu về kích thước và cấu trúc trong các bài toán điều khiển đối với những hệ thống lớn và phức tạp, các phương pháp điều khiển truyền thống sử dụng một bộ điều khiển tập trung trở nên không thực tế, thậm chí bất khả thi trong những trường hợp có nhiều sự tương tác lẫn nhau giữa các hệ con. Điều này khơi lại sự chú ý và thúc đẩy nghiên cứu về điều khiển phân tán dành cho các hệ thống lớn và phức tạp.

Luận văn này nhằm mục tiêu cung cấp các công cụ thiết kế những bộ điều khiển phân tán dự đoán dựa trên mô hình, dành cho các hệ thống tuyến tính. Đặc trưng của phương pháp điều khiển dự đoán dựa trên mô hình (Model Predictive Control - MPC) là trình bày bài toán điều khiển dưới dạng một bài toán tối ưu hóa sẽ được giải đi giải lại trong từng khoảng thời gian ngắn. Điều này cho phép đơn giản hóa việc thiết kế bộ điều khiển dành cho các hệ thống nhiều ngõ ra nhiều ngõ vào (multi-input multi-output systems) với những ràng buộc cứng, giúp cho phương pháp này được chấp nhận rộng rãi trong công nghiệp. Tuy nhiên, đối với các hệ thống lớn và phức tạp, cách thức này làm nảy sinh những vấn đề khó: cần có một cơ chế truyền thông tin để gửi tất cả các số liệu đo đạc về một mối; độ phức tạp của bài toán tối ưu đòi hỏi nhiều thời gian tính toán; cả hệ thống phụ thuộc vào kết quả tạo ra bởi một bộ xử lý trung tâm, nên giảm độ linh hoạt và bền vững của hệ thống. Do vậy, để thiết kế các bộ điều khiển phân tán dự đoán dựa trên mô hình, luận văn này tập trung vào việc giải bài toán tối ưu hóa kích thước lớn của phương pháp điều khiển MPC bằng phương pháp phân tán.

Cụ thể, luận văn này giải quyết các vấn đề sau:

1. Thiết kế các thuật toán phân tán để giải bài toán tối ưu lồi trong điều khiển MPC dành cho các hệ thống tuyến tính rời rạc.

2. Xác lập các điều kiện để đạt được các tính chất chấp nhận được (feasibility) và ổn định (stability) của hệ thống điều khiển vòng kín.

Thiết lập của bài toán điều khiển bao gồm một nhóm các bộ điều khiển địa phương gắn với các hệ con có khả năng liên lạc hạn chế với nhau, và mỗi bộ điều khiển có một bộ xử lý để đảm nhận công việc tính toán cục bộ. Bài toán tối ưu hóa MPC ban đầu được trình bày dưới dạng một bài toán tối ưu bình phương lồi, gồm có một hàm mục tiêu lồi bậc hai phân rã được và một số ràng buộc tuyến tính, trong đó mỗi ràng buộc chỉ liên hệ tới một số lượng nhỏ các hệ con, nghĩa là có sự móc nối thưa thớt được quy định bởi các ràng buộc. Để giải bài toán dạng này theo phương pháp phân tán, các kỹ thuật phân rã đối ngẫu (dual decomposition) được sử dụng, nhờ đó thu được những bài toán tương đương trong không

gian đối ngẫu có tính chất dễ chia tách. Bằng việc định nghĩa các biến số địa phương (gồm có biến trạng thái và lệnh điều khiển) và khu vực láng giềng của một hệ con (gồm các hệ con có thể tương tác và liên lạc trực tiếp với hệ con đã định), các thuật toán sử dụng đạo hàm bậc nhất có thể được triển khai một cách phân tán để giải bài toán đối ngẫu. Chúng tôi đề xuất ba thuật toán phân tán (distributed) và phân cấp (hierarchical) chính: thuật toán phân tán Han, thuật toán phân tán tăng tốc dùng gần-đạo hàm, và thuật toán phân cấp cho nghiệm khả thi nguyên thủy sử dụng đạo hàm trong không gian đối ngẫu.

Đầu tiên, phương pháp đối ngẫu Fenchel được dùng để xây dựng hàm số đối ngẫu, trong đó hàm số chỉ thị tập hợp (indicator function) được dùng để nới lỏng các ràng buộc. Giải thuật cơ sở là thuật toán song song của Han, thuộc về lớp các thuật toán chiếu sử dụng đạo hàm. Chúng tôi chỉ ra rằng các bài toán con của phương pháp Han có các nghiệm dạng biểu thức, và vì thế thuật toán chỉ bao gồm vòng lặp các phép tính đại số tuyến tính, chúng vừa dễ và vừa có thể được triển khai với một thiết lập phân tán. Phương pháp phân tán Han thu được cũng được chứng minh là sản sinh kết quả tương đồng với kết quả của phương pháp tập trung tương ứng, do vậy cũng hội tụ về nghiệm của bộ điều khiển MPC tập trung đối với mỗi bước lấy mẫu. Dựa trên sự hội tụ của thuật toán, tính chấp nhận được và ổn định của hệ thống vòng kín được kế thừa từ thiết lập MPC tập trung.

Tiếp theo, một thuật toán phân tán tăng tốc dùng gần-đạo hàm được dùng để giải bài toán đối ngẫu, cái có thể thu được từ phương pháp phân rã đối ngẫu Lagrange hoặc Fenchel. Giải thuật này thuộc về lớp thuật toán dùng đạo hàm bậc nhất có tăng tốc, vốn được biết có tốc độ hội tụ nhanh nhất trong tất cả các thuật toán dùng đạo hàm bậc nhất. Chúng tôi chỉ ra rằng giải thuật tăng tốc này có thể được xem như một phiên bản mở rộng và nâng cao của thuật toán Han, vì nó hội tụ nhanh hơn một bậc so với thuật toán dùng gần-đạo hàm kinh điển, thực chất tương đương với phương pháp Han khi xét trên các bài toán tối ưu bình phương. Hơn nữa, bằng việc sử dụng hàm số chỉ thị tập hợp, chúng tôi có thể giải quyết một bài toán với hàm mục tiêu trộn lẫn các chuẩn bậc nhất và bậc hai, bằng cách xây dựng được hàm số đối ngẫu khả vi dù cho bài toán ban đầu không khả vi. Bởi công việc tính toán cần thêm nhằm mục tiêu tăng tốc chỉ là một phép tính kết hợp tuyến tính của các nghiệm thu được ở hai bước lặp gần nhất, thuật toán tăng tốc này chỉ sử dụng thêm một ít bộ nhớ để lưu các giá trị biến số cũ, trong khi khối lượng tính toán cũng ít gần như là phương pháp kinh điển tương ứng. Do vậy, thuật toán này có thể được triển khai phân tán với một cách tương tự như thuật toán phân tán Han.

Trong phương pháp thứ ba, một cách tiếp cận phân cấp với hai vòng lặp được dùng để giải bài toán đối ngẫu Lagrange của bài toán tối ưu hóa lồi MPC tập trung. Trong lớp vòng lặp bên ngoài, hàm số đối ngẫu được cực đại hóa bằng phương pháp chiếu dùng đạo hàm kết hợp với việc lấy bình quân nhằm cung cấp các giới hạn đối với sự vi phạm tính chấp nhận được và mức độ dưới tối ưu của hàm số nguyên thủy. Trong lớp vòng lặp bên trong, một thuật toán tối ưu hóa phân cấp được dùng để tạo ra một nghiệm chính xác hoặc gần đúng với độ chính xác tùy ý của bài toán cực tiểu hóa hàm Lagrangian. Chúng tôi trình bày hai giải thuật dành cho vòng lặp bên trong: một cái là phương pháp đạo hàm liên hợp phân cấp, và cái kia là giải thuật tối ưu hóa phân tán kiểu Jacobi. Phương pháp này có thể được áp dụng đối với các bài toán MPC đạt điều kiện có tồn tại nghiệm trong bước lấy mẫu đầu tiên và thỏa điều kiện Slater (tức là tồn tại một nghiệm thỏa mãn nghiêm ngặt các ràng buộc dạng bất đẳng thức). Sử dụng phương pháp này, bộ điều khiển có thể tạo ra các nghiệm chấp nhận được của bài toán MPC ngay cả trong khi nghiệm của bài toán đối ngẫu chưa đạt điểm tối ưu, và đạt tính ổn định của hệ thống vòng kín.

Bên cạnh việc xây dựng các thuật toán mới, luận văn này cũng chú trọng các vấn đề về mặt triển khai qua việc xem xét một ứng dụng trong điều khiển nhà máy thủy điện. Chúng tôi nghiên cứu bài toán điều khiển đối với một thung lũng thủy điện (hydro power valley) với mô hình phi tuyến. Một số chủ đề đã được xem xét, bao gồm phương pháp giảm bậc mô hình và cách thiết lập lại bài toán tối ưu hóa MPC sao cho bài toán tối ưu hóa thu được là phù hợp để áp dụng các thuật toán phân tán đã được phát triển trong luận văn này. Chúng tôi chỉ ra rằng khi triển khai thuật toán phân tán tăng tốc dùng gần-đạo hàm của chúng tôi, bộ điều khiển phân tán đạt được hiệu năng tốt tương đương với một bộ điều khiển tập trung, trong khi đó thuật toán phân tán chỉ sử dụng thời gian tính toán của CPU ít hơn hẳn so với một chương trình giải tập trung. Các kết quả từ ví dụ ứng dụng này xác nhận và cổ vũ việc áp dụng phương pháp điều khiển phân tán dự đoán dựa trên mô hình đối với các hệ thống lớn và phức tạp.

Contents

Chapter 1

Introduction

This chapter provides the motivation for the research and a survey of the state-of-the-art on distributed model predictive control. We also present the background of the problems that serve as a starting point for the subsequent chapters. Research objectives and summary of contributions are provided, followed by an overall outline of the thesis.

1.1 Motivation and literature survey

Nowadays, Model Predictive Control (MPC) is widely used for controlling industrial processes (Qin and Badgwell, 2003), and its properties and design considerations have also been studied thoroughly by the scientific community (Maciejowski, 2002, Mayne et al., 2000, Rawlings and Mayne, 2009). The essence of MPC is to solve online an optimization problem that captures the outcomes of the predicted control sequence in a receding horizon fashion. MPC can naturally handle operational constraints and, moreover, it is designed for multi-input multi-output systems, both of which contributed to the popularity of MPC. Moreover, since MPC relies on optimization techniques to solve the control problem, improvements in optimization techniques can help to broaden the application of MPC for more complex problems.

When considering a control problem for a large-scale networked system (such as complex manufacturing or infrastructure processes), using MPC in a centralized fashion may become impractical and unsuitable due to the computational burden and the requirement of global communications across the network. Centralized MPC is also inflexible against the limitation of information exchange between different authorities controlling the local subsystems and against changes in the network structure, i.e, the whole centralized model needs to be updated for every small change. In order to deal with these limitations, Distributed MPC (DMPC) has been proposed for control of such large-scale systems, by decomposing the overall system into several small subsystems (Jia and Krogh, 2001, Camponogara et al., 2002). The subsystems then employ separate MPC controllers that only solve local control problems, use local information from neighboring subsystems, and collaborate to achieve globally attractive solutions.

DMPC is an emerging topic for scientific research. The open issues of DMPC have recently been discussed by Rawlings and Stewart (2008), Scattolini (2009). Several DMPC methods were proposed for different problem setups. For systems with decoupled dynamics, a DMPC scheme for multiple vehicles with coupled cost functions was proposed by Dunbar and Murray (2006), utilizing predicted trajectories of the neighbors in each subsystem's optimization. A DMPC scheme with a sufficient stability test for dynamically decoupled systems was presented by Keviczky et al. (2006), in which each subsystem optimizes also over the behaviors of its neighbors. Richards and How (2007) proposed a robust DMPC method to deal with disturbances in the setting with decoupled systems and coupled constraints, based on constraint tightening and a serial solution approach. For systems with coupled dynamics and decoupled constraints, a DMPC scheme has been developed by Venkat et al. (2008) based on a Jacobi algorithm that deals with the primal problem, using a convex combination of new and old solutions. In (Jia and Krogh, 2002), the neighboring subsystem states are treated as bounded contracting disturbances, and each subsystem solves a min-max problem. A partitioning-based algorithm was proposed by Alessio and Bemporad (2007, 2008), with sufficient conditions for the a posteriori stability analysis. Li et al. (2005) proposed an algorithm with stability conditions in which subproblems are solved in parallel in order to get a Nash equilibrium. Several DMPC algorithms based on decomposion of the global optimization problems were proposed by Camponogara and Talukdar (2007), Necoara et al. (2008), Necoara and Suykens (2008). Other recent work on applications of DMPC is reported in (Mercangoz and Doyle III, 2007, Negenborn et al., 2009, Arnold et al., 2010).

Throughout this thesis, we will tackle the MPC optimization problem using dual decomposition approaches. Dual decomposition is a well-established concept since around 1960 when Uzawa's algorithm (Arrow et al., 1958) was presented. Similar ideas were exploited in large-scale optimization (Danzig and Wolfe, 1961). Over the next decades, methods for decomposition and coordination of dynamic systems were developed and refined (Findeisen, 1980, Mesarovic et al., 1970, Singh and Titli, 1978) and used in large-scale applications (Carpentier and Cohen, 1993). In (Tsitsiklis et al., 1986) a distributed asynchronous method for solving the dual problem was studied. More recently dual decomposition has been applied in the DMPC literature in (Doan et al., 2011c, 2009, Giselsson and Rantzer, 2010, Negenborn et al., 2008) for problems with a strongly convex quadratic cost and arbitrary linear constraints.

1.2 Distributed MPC settings and problems

1.2.1 Subsystems and their neighborhood

Consider a plant consisting of M dynamically coupled subsystems. The dynamics of each subsystem are assumed linear and to be influenced directly by only a *small* number of other subsystems. Moreover, each subsystem i is assumed to have local linear coupled constraints involving only variables from a small number of other subsystems.

Based on the couplings, we define the 'neighborhood' of subsystem i, denoted as $\mathcal{N}^i$, as the set including i and the indices of subsystems that have either a direct dynamical coupling or a constraint coupling with subsystem i. In Figure 1.1, we demonstrate this

with a diagram where each node stands for one subsystem, the dotted links show constraint couplings, and the solid links represent dynamical couplings.

1.2.2 Coupled subsystem model

We assume that each subsystem can be represented by a discrete-time, linear time-invariant model of the form[1]:

$$x_{k+1}^i = \sum_{j \in \mathcal{N}^i} A^{ij} x_k^j + B^{ij} u_k^j, \tag{1.1}$$

where $x_k^i \in \mathbb{R}^{n^i}$ and $u_k^i \in \mathbb{R}^{m^i}$ are the states and control inputs of the i-th subsystem at time step k, respectively.

Denoting the aggregate state and input variables by x and u, the centralized system dynamics is:

$$x_{k+1} = Ax_k + Bu_k \tag{1.2}$$

with $x_k = [(x_k^1)^T (x_k^2)^T \dots (x_k^M)^T]^T$, $u_k = [(u_k^1)^T (u_k^2)^T \dots (u_k^M)^T]^T$, $A = [A^{ij}]_{i,j \in \{1,\dots,M\}}$ and $B = [B^{ij}]_{i,j \in \{1,\dots,M\}}$, where $A^{ij} = 0$ and $B^{ij} = 0$ for $j \notin \mathcal{N}^i$.

1.2.3 Linear coupled constraints

Each subsystem i is assumed to have local linear coupled constraints involving only variables within its neighborhood $\mathcal{N}^i$. Within one prediction period, all constraints that subsystem i is involved in can be written in the following form:

$$\sum_{j \in \mathcal{N}^i} \sum_{k=0}^{N-1} D_k^{ij} x_k^j + E_k^{ij} u_k^j = c_{\text{eq}} \tag{1.3}$$

$$\sum_{j \in \mathcal{N}^i} \sum_{k=0}^{N-1} \bar{D}_k^{ij} x_k^j + \bar{E}_k^{ij} u_k^j \leq \bar{c}_{\text{ineq}} \tag{1.4}$$

in which N is the prediction horizon, c_{eq} and $\bar{c}_{\text{ineq}}$ are column vectors, and D_k^{ij}, E_k^{ij}, $\bar{D}_k^{ij}$, and $\bar{E}_k^{ij}$ are matrices with appropriate dimensions.

1.2.4 Formulation of the centralized MPC problem

We will formulate the centralized MPC problem for systems of the form (1.1) using a convex quadratic cost function and linear constraints, with an additional terminal point constraint to zero out all terminal states. Under the conditions that a feasible solution of the centralized MPC problem exists, and that the point with zero states and inputs is in

[1] This system description with only the state update equation is chosen for simplicity of exposition, our framework can also be extended to output-feedback distributed MPC for observable systems.

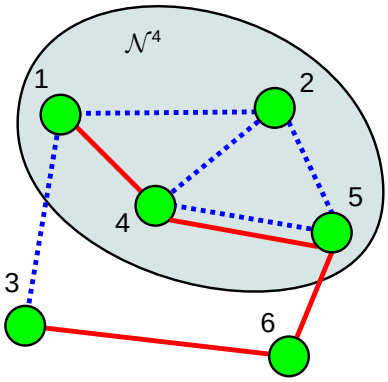


Figure 1.1: *A figure showing the constraint couplings (dotted links) and dynamical couplings (solid links) between subsystems. In this example,* $\mathcal{N}^4 = \{4, 1, 2, 5\}$.

the relative interior of the constraint set, this MPC scheme ensures feasibility and stability, as shown by Mayne et al. (2000) and Keerthi and Gilbert (1988). However, the algorithm proposed in this paper will also work with any other centralized MPC approach that does not require a terminal point constraint, provided that the subsystems have local stabilizing terminal controllers. We will further assume without loss of generality that the initial time is zero.

The optimization variable of the centralized MPC problem is constructed as a stacked vector of predicted subsystem control inputs *and* states that affect the predicted states from step 1 to step N that is the prediction horizon:

$$\begin{aligned}\mathbf{x} = \Big[&\left(u_0^1\right)^T, \ldots, \left(u_0^M\right)^T, \ldots, \left(u_{N-1}^1\right)^T, \ldots, \left(u_{N-1}^M\right)^T, \\ &\left(x_1^1\right)^T, \ldots, \left(x_1^M\right)^T, \ldots, \left(x_N^1\right)^T, \ldots, \left(x_N^M\right)^T\Big]^T\end{aligned} \tag{1.5}$$

Recall that n^i and m^i denote the numbers of states and inputs of subsystem i. The number of optimization variables for the centralized problem is thus:

$$n_{\mathbf{x}} = N\sum_{i=1}^{M} m^i + N\sum_{i=1}^{M} n^i \tag{1.6}$$

The cost function of the centralized MPC problem is assumed to be decoupled and convex quadratic:

$$J = \sum_{i=1}^{M}\sum_{k=0}^{N-1}\left(\left(u_k^i\right)^T R_i u_k^i + \left(x_{k+1}^i\right)^T Q_i x_{k+1}^i\right) \tag{1.7}$$

with positive definite weights R_i, Q_i. This cost function can be rewritten using the decision

variable $\mathbf{x}$ as

$$J = \mathbf{x}^T H \mathbf{x} \tag{1.8}$$

in which the Hessian H is a positive definite matrix of the following form:

$$H = \begin{bmatrix} R & 0 \\ 0 & Q \end{bmatrix} \tag{1.9}$$

where R and Q are block-diagonal, positive definite weights and are built from R_i and Q_i as follows:

$$\begin{aligned} R &= \mathrm{diag}(\underbrace{\tilde{R}, \ldots, \tilde{R}}_{N \text{ times}}) \quad \text{with } \tilde{R} = \mathrm{diag}(R_1, \ldots, R_M) \\ Q &= \mathrm{diag}(\underbrace{\tilde{Q}, \ldots, \tilde{Q}}_{N \text{ times}}) \quad \text{with } \tilde{Q} = \mathrm{diag}(Q_1, \ldots, Q_M) \end{aligned}$$

Remark 1.1 *The positive definiteness assumption on Q_i and R_i and the choice of the centralized variable as described in (1.5) without eliminating state variables will help to compute the inverse of the Hessian easily, by allowing simple inversion of each block on the diagonal of the Hessian.*

The centralized MPC problem, denoted by $(\mathcal{P})$, is defined as follows:

$$\min_{\substack{u_0^1 \ldots u_{N-1}^M, \\ x_1^1 \ldots x_N^M}} \quad \sum_{i=1}^{M} \sum_{k=0}^{N-1} \left(\left(u_k^i\right)^T R_i u_k^i + \left(x_{k+1}^i\right)^T Q_i x_{k+1}^i \right) \tag{1.10}$$

$$\text{s.t.} \quad x_{k+1}^i = \sum_{j \in \mathcal{N}^i} A^{ij} x_k^j + B^{ij} u_k^j, \quad i = 1, \ldots, M, \quad k = 0, \ldots, N-1 \tag{1.11}$$

$$x_N^i = 0, \quad i = 1, \ldots, M \tag{1.12}$$

$$\sum_{j \in \mathcal{N}^i} \sum_{k=0}^{N-1} D_k^{ij} x_k^j + E_k^{ij} u_k^j = c_{\mathrm{eq}}, \quad i = 1, \ldots, M \tag{1.13}$$

$$\sum_{j \in \mathcal{N}^i} \sum_{k=0}^{N-1} \bar{D}_k^{ij} x_k^j + \bar{E}_k^{ij} u_k^j \leq \bar{c}_{\mathrm{ineq}}, \quad i = 1, \ldots, M \tag{1.14}$$

1.2.5 Distributed MPC problem

Our approach for distributed MPC is to solve the centralized optimization problem (1.10)–(1.14) in a distributed manner which then also leads to a distributed or hierarchical control architecture. In the subsequent chapters, we propose and investigate different distributed

MPC methods that are developed based on distributed optimization algorithms. This approach has several advantages: firstly, we can make use of state-of-the-art distributed optimization algorithms that are actively developed not only in the field of control but also in other fields such as signals and image processing, machine learning, and large-scale and cluster computing. Another advantage is the possibility to achieve centralized optimality with distributed methods; thus the distributed MPC will inherit the properties of the corresponding centralized MPC method, e.g., stability or feasibility of the MPC. In case a distributed algorithm does not achieve centralized optimality, we can still quantify the sub-optimality of the distributed solution with respect to the global optimum. Last but not least, designing distributed MPC based on distributed optimization often leads to a systematic way of defining subsystems and sub-problems, thus simplifying the implementation of distributed model predictive controllers.

1.3 Research objectives

With the focus on distributed MPC design that is guided by distributed optimization methods, the research presented in this thesis is aimed at the following topics:

- Investigate coordination and distribution methods that lead to distributed algorithms for solving optimization problems arising in MPC of large-scale systems, which often have sparse structures.
- Design distributed and hierarchical MPC methods that provide feasible control actions that lead to closed-loop stability of the whole system.
- Compare analytically and numerically the performance of centralized and distributed MPC solutions when they are implemented in the same control problem.
- Find the formulation of MPC problems and the construction of subsystems so that distributed MPC can be implemented efficiently.

These topics will be tackled independently or simultaneously in each of the Chapters 2, 3, 4, and 5.

1.4 Summary of contributions

The contributions of this thesis in the field of distributed and hierarchical MPC can be summarized as follows:

1. We develop a distributed optimization method for strictly convex quadratic programs. This distributed algorithm is able to generate the same iterates as the parallel (centralized) counterpart that was presented in Han and Lou (1988), and is guaranteed to converge to the centralized solution. Materials related to this work have been reported in Doan et al. (2009, 2010, 2011c).

2. We develop a distributed version of the accelerated proximal gradient method (Beck and Teboulle, 2009) that can handle optimization problems with mixed 1-norm and 2-norm penalty terms in the cost function and under linear constraints with a sparse coupling structure. The distributed accelerated proximal gradient algorithm can also be considered as an extension of the distributed Han's algorithm and achieves the best convergence rate within the class of gradient methods. This work has been reported in Giselsson et al..

3. We propose a constraint tightening approach that leads to two hierarchical algorithms for solving the MPC optimization problem based on dual decomposition. These algorithms can be terminated after a finite number of iterations, but still provide feasible solutions to the MPC problem. The algorithms also guarantee a monotonic decrease of the cost function, thus leading to MPC stability. The two hierarchical algorithms have been reported in Doan et al. (2011b,a).

4. We present a complete treatment of a hydro power valley system, using nonlinear modeling with linear approximation and model order reduction, and then design an efficient distributed MPC controller that can obtain a performance that is comparable with that of centralized MPC. In the problem formulation, we will propose a systematic method to handle the coupling that appears in the cost function, which results in a sparse problem that is favorable for distributed MPC. This work has been reported in Doan et al. (2012).

1.5 Thesis outline

The material presented in this thesis is organized as separate chapters, in the following order:

- Chapter 2: We investigate Han's parallel method for convex quadratic problems, and then design a distributed version of this method, which can generate equivalent iterates resulting from the centralized algorithm. Stability of the closed-loop MPC is guaranteed as an inheritance from the centralized MPC that is based on feasibility, which is indeed obtained upon convergence of the algorithm.

- Chapter 3: We analyze the classical proximal gradient method and show its similarity to Han's parallel method. This method is outperformed by the accelerated proximal gradient algorithm, which achieves the best convergence rate for a gradient algorithm that solves a strongly convex optimization problem. We derive a distributed MPC scheme based on the accelerated proximal gradient algorithm.

- Chapter 4: We tackle the asymptotic convergence of dual decomposition approaches, which often do not provide a feasible solution of the primal optimization problem in a finite number of iterations. The main idea is to use a constraint tightening approach, and then apply a primal-dual iterative algorithm that provides bounds on the constraint violation and the sub-optimality. We develop two primal-dual iterative algorithms, leading to two hierarchical MPC methods that provide feasible solutions and achieve closed-loop stability.

- Chapter 5: In order to evaluate the efficiency of distributed MPC, we apply the distributed MPC approach based on the accelerated proximal gradient method to the control problem of a hydro power valley system. The simulation results show that distributed MPC can achieve nearly the same performance that is produced by centralized MPC, while the computation time for distributed MPC is remarkably lower than for centralized MPC.
- Chapter 6: Summary of the results, and recommendations for future research.

Below is the table showing connections among the chapters, the arrows suggest the order for reading the chapters that could help in understanding the links between chapters, however each chapter also provides complete details of the topic and can be read alone.

Distributed MPC based on solving the underlying optimization problem exactly	Chapter 2 ↓ Chapter 3 ↓ Chapter 5
Hierarchical MPC based on suboptimal solution of the underlying optimization problem	Chapter 4

Chapter 2

Distributed Han's method for convex quadratic problems

We investigate Han's parallel method for convex quadratic problems, and then design a distributed version of this method, which can generate equivalent iterates as those resulted from the centralized algorithm. The underlying decomposition technique relies on Fenchel's duality and allows subproblems to be solved using local communications only. Further, we propose two techniques aimed at improving the convergence rate of the iterative approach and illustrate the results using a numerical example. We conclude by discussing open issues of the proposed method and by providing an outlook on research in the field.

2.1 Introduction

In this chapter, we present a decomposition scheme based on Han's parallel method (Han and Lou, 1988), aiming to solve the centralized optimization problem of MPC (1.10)–(1.14) in a distributed way. This approach results in two distributed algorithms that are applicable to DMPC of large-scale industrial processes. The main ideas of our algorithms are to find a distributed update method that is equivalent to Han's method (which relies on global communications), and to improve the convergence speed of the algorithm. We demonstrate the application of our methods in a simulated water network control problem. The open issues of the proposed scheme will be discussed to formulate future research directions.

This chapter is organized as follows. In Section 2.2, we summarize Han's parallel method for convex programs (Han and Lou, 1988) as the starting point for our approach. In Section 2.3, we present two distributed MPC schemes that solve the dual optimization problem by using only local communications. The first DMPC scheme uses a distributed iterative algorithm that we prove to generate the same iterates as Han's algorithm. As a consequence of this equivalence, the proposed DMPC scheme achieves the global optimum upon convergence and thus inherits feasibility and stability properties from its centralized MPC

counterpart. The second DMPC scheme is an improved algorithm that aims to speed up the convergence of the distributed approach. In Section 2.4, we illustrate the application of the new DMPC schemes in an example system involving irrigation canals. In Section 2.5, we discuss the open issues of Han's method that motivate directions for future research. Section 2.6 concludes the chapter.

Before we go into the details, we first rewrite the problem (1.10)–(1.14) in a compact form as

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{x}^T H \mathbf{x} \\ \text{s.t.} \quad & a_l^T \mathbf{x} = b_l, \quad l = 1, \ldots, n_{\text{eq}} \\ & a_l^T \mathbf{x} \leq b_l, \quad l = n_{\text{eq}} + 1, \ldots, s \end{aligned} \tag{2.1}$$

with $s = n_{\text{eq}} + n_{\text{ineq}}$, and where the matrix H is positive definite and block-diagonal due to the formulation of the centralized MPC problem (1.10)–(1.14). The algorithms to be described in the next sections will focus on how to solve this optimization problem in a distributed way.

2.2 Han's parallel method for convex programs

Han's algorithm (Han and Lou, 1988) is a method to decompose Fenchel's dual problem (Rockafellar, 1970) of a convex optimization problem. Fenchel's duality theorem aims at minimizing a difference $f(\mathbf{x}) - g(\mathbf{x})$, where f is a convex function and g is a concave function. A special case of this problem is minimizing f over a convex constraint set C, where g is a penalty function for violating the constraint. In Han's problem, the set C is the intersection of many constraint sets, and the dual variables are iteratively projected onto the local constraint sets. As a consequence, the sum of the dual variables converges to the minimizer of the Fenchel's dual problem (Han and Lou, 1988). In this section, we summarize the main elements of Han's parallel method, followed by a simplified version for the case of positive definite quadratic programming.

2.2.1 Han's algorithm for general convex problems

The class of optimization problems tackled by Han's algorithm is the following:

$$\begin{aligned} \min_{\mathbf{x}} \quad & q(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in C \triangleq C_1 \cap \cdots \cap C_s \end{aligned} \tag{2.2}$$

where $C_1, \cdots, C_s$ are closed convex sets and $C \neq \emptyset$, and where $q(\mathbf{x})$ is uniformly convex[1] and differentiable on $\mathbb{R}^{n_{\mathbf{x}}}$.

[1] A function $q : \mathbb{R}^n \to \mathbb{R}$ is uniformly convex (or strongly convex) on a set $S \subset \mathbb{R}^n$ if there is a constant $\rho > 0$ such that for any $\mathbf{x}_1$, $\mathbf{x}_2 \in S$ and for any $\lambda \in (0, 1)$:

$$q(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) \leq \lambda q(\mathbf{x}_1) + (1-\lambda) q(\mathbf{x}_2) - \rho\lambda(1-\lambda)\|\mathbf{x}_1 - \mathbf{x}_2\|^2.$$

Problems of type (2.2) can be solved by Han's algorithm. In the following algorithm we will describe Han's method, which is an iterative procedure. We use p as iteration counter of the algorithm, and the superscript (p) for variables that are computed at iteration p.

Algorithm 2.1 ***Han's algorithm for convex programs***

Let α be a sufficiently large number[2]*, define the vectors $\mathbf{y}^{(p)}, \mathbf{y}_l^{(p)} \in \mathbb{R}^{n_\mathbf{x}}, l = 1, \ldots, s$ with p the iteration index, and set $\mathbf{y}^{(0)} = \mathbf{y}_1^{(0)} = \cdots = \mathbf{y}_s^{(0)} = 0$. Also set $\mathbf{x}^{(0)} = \nabla q^* \left(\mathbf{y}^{(0)}\right)$ with q^* being the conjugate function*[3] *of q. For $p = 1, 2, \ldots,$ we perform the following computations:*

1) For $l = 1, \ldots, s$, find $\mathbf{z}_l^{(p)}$ that solves

$$\begin{aligned} \min_{\mathbf{z}} \quad & \phi(\mathbf{z}) \triangleq \frac{1}{2} \left\| \mathbf{z} + \alpha \mathbf{y}_l^{(p-1)} - \mathbf{x}^{(p-1)} \right\|_2^2 \\ \text{s.t.} \quad & \mathbf{z} \in C_l \end{aligned} \tag{2.3}$$

2) Assign

$$\mathbf{y}_l^{(p)} = \mathbf{y}_l^{(p-1)} + (1/\alpha) \left(\mathbf{z}_l^{(p)} - \mathbf{x}^{(p-1)} \right) \tag{2.4}$$

3) Set $\mathbf{y}^{(p)} = \mathbf{y}_1^{(p)} + \cdots + \mathbf{y}_s^{(p)}$

4) Compute

$$\mathbf{x}^{(p)} = \nabla q^* \left(\mathbf{y}^{(p)} \right) \tag{2.5}$$

Han and Lou (1988) also showed that Algorithm 2.1 converges to the global optimum under the given conditions on q and C, i.e., $q(\cdot)$ is uniformly convex and $C \neq \emptyset$ is the intersection of closed convex sets.

Remark 2.1 *Han's method essentially solves the dual problem of (2.2), so that $\mathbf{y}^{(p)}$ converges to the solution of the Fenchel's dual problem:*

$$\min_{\mathbf{y} \in \mathbb{R}^{n_\mathbf{x}}} \left(q^*(\mathbf{y}) + \delta^*(\mathbf{y} | -C) \right) \tag{2.6}$$

[2] α is a design parameter that has to be sufficiently large. With $\alpha \geq s/\rho$ Han's method will converge (Han and Lou, 1988). For positive definite QPs we can choose ρ as one half of the smallest eigenvalue of the Hessian matrix.

[3] The conjugate function of a function $q : \mathbb{R}^n \to \mathbb{R}$ is defined by: $q^*(\mathbf{y}) = \sup_{\mathbf{x} \in \mathbb{R}^n} \left(\mathbf{y}^T \mathbf{x} - q(\mathbf{x}) \right)$. The conjugate function q^* is always convex (Boyd and Vandenberghe, 2004). This formulation is called *convex conjugate function* in Rockafellar (1970), in order to distinguish it with the *concave conjugate function*, defined by $q_*(\mathbf{y}) = \inf_{\mathbf{x} \in \mathbb{R}^n} \left(\mathbf{y}^T \mathbf{x} - q(\mathbf{x}) \right)$.

in which $\delta(\mathbf{x}|C)$ is the indicator function, which is 0 if $\mathbf{x} \in C$ and ∞ otherwise. The conjugate function of $\delta(\mathbf{x}|C)$ is $\delta^*(\mathbf{y}|C) = \sup_{\mathbf{x}\in C} \mathbf{y}^T\mathbf{x}$. According to Fenchel's duality theorem (Rockafellar, 1970), the minimum of the convex function $f(\mathbf{x}) - g(\mathbf{x})$, where f is a convex function on $\mathbb{R}^{n\times}$ and g is a concave function on $\mathbb{R}^{n\times}$, equals the maximum of the concave function $g_*(\mathbf{y}) - f^*(\mathbf{y})$, or equivalently the minimum of $f^*(\mathbf{y}) - g_*(\mathbf{y})$. In this situation $f \equiv q$ and $g \equiv -\delta$, and note that $-\delta_*(\mathbf{y}|C) = -\inf_{\mathbf{x}\in C} \mathbf{y}^T\mathbf{x} = \sup_{-\mathbf{x}\in C}(\mathbf{y}^T(-\mathbf{x})) = \delta^*(\mathbf{y}|-C)$. The value $\mathbf{y}^{(\bar{p})}$ achieved when Algorithm 2.1 converges is an optimizer of (2.6), and hence $\mathbf{x}^{(\bar{p})} = \nabla q^*\left(\mathbf{y}^{(\bar{p})}\right)$ is the solution of (2.2).

Remark 2.2 *The optimization (2.3) aims to find the projection of $\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}$ on the set C_l, and its dual problem is*

$$\min_{\mathbf{y}} \left\{ (\alpha/2)\|\mathbf{y} - \mathbf{y}_l^{(p-1)}\|_2 + {\mathbf{x}^{(p-1)}}^T\mathbf{y} + \delta^*(\mathbf{y}| - C_l) \right\} \tag{2.7}$$

The update formula (2.4) gives $\mathbf{y}_l^{(p)} = \nabla\phi\left(\mathbf{z}_l^{(p)}\right)$, where ϕ is the cost function of (2.3). Fenchel's duality also guarantees that $\mathbf{y}_l^{(p)} = \nabla\phi(\mathbf{z}_l^{(p)})$ is the solution of (2.7), since $\mathbf{z}_l^{(p)}$ is the solution of (2.3).

The uniform convexity of q is used to derive the inequality:

$$(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})^T(\mathbf{y}^{(k)} - \mathbf{y}^{(k-1)}) \leq \frac{s}{2\rho}\sum_{l=1}^{s} \|\mathbf{y}_l^{(k)} - \mathbf{y}_l^{(k-1)}\|_2^2 \tag{2.8}$$

and then (2.8) is used to prove the inequality

$$q^*(\mathbf{y}^{(k-1)}) + \delta^*(\mathbf{y}^{(k-1)}| - C) \geq q^*(\mathbf{y}^{(k)}) + \delta^*(\mathbf{y}^{(k)}| - C) + \frac{\alpha\rho - s}{2\rho}\sum_{l=1}^{s} \|\mathbf{y}_l^{(k)} - \mathbf{y}_l^{(k-1)}\|_2^2 \tag{2.9}$$

By applying (2.9) successively with $k = 1, \ldots, p$ we get

$$\left\{ q^*(\mathbf{y}^{(0)}) + \delta^*(\mathbf{y}^{(0)}| - C) \right\} - \left\{ q^*(\mathbf{y}^{(p)}) + \delta^*(\mathbf{y}^{(p)}| - C) \right\} \geq \frac{\alpha\rho - s}{2\rho}\sum_{k=1}^{p} \left\{ \sum_{l=1}^{s} \|\mathbf{y}_l^{(k)} - \mathbf{y}_l^{(k-1)}\|_2^2 \right\} \tag{2.10}$$

The left hand side of (2.10) is bounded below, and the right hand side of (2.10) is nonnegative since $\alpha\rho - s \geq 0$, thus $\sum_{l=1}^{s} \|\mathbf{y}_l^{(k)} - \mathbf{y}_l^{(k-1)}\|_2^2 \to 0$ as $p \to \infty$. This leads to the convergence of $\|\mathbf{y}^{(p)} - \mathbf{y}^{(p-1)}\|_2 \to 0$ and $\|\mathbf{x}^{(p)} - \mathbf{x}^{(p-1)}\|_2 \to 0$ as $p \to \infty$.

2.2.2 Han's algorithm for positive definite quadratic programs

In case the optimization problem has a positive definite cost function and linear constraints as in (2.2), the optimization problem (2.3) and the derivative of conjugate function (2.5)

have analytical solutions, and then Han's method becomes simpler. In the following we show how the analytical solutions of (2.3) and (2.5) can be obtained when applying Algorithm 2.1 to the problem (2.1).

Remark 2.3 *The result of simplifying Han's method in this section is slightly different from the original one described in Han and Lou (1988), so as to correct the minor mistakes we found in that paper.*

As in (2.2), each constraint $\mathbf{x} \in C_l$ is implicitly expressed by a scalar linear equality or inequality constraint. So (2.3) takes one of the following two forms:

$$\begin{aligned} \min_{\mathbf{z}} \quad & \frac{1}{2}\|\mathbf{z} + \alpha\mathbf{y}_l^{(p-1)} - \mathbf{x}^{(p-1)}\|_2^2 \\ \text{s.t.} \quad & a_l^T\mathbf{z} = b_l \end{aligned} \tag{2.11}$$

or

$$\begin{aligned} \min_{\mathbf{z}} \quad & \frac{1}{2}\|\mathbf{z} + \alpha\mathbf{y}_l^{(p-1)} - \mathbf{x}^{(p-1)}\|_2^2 \\ \text{s.t.} \quad & a_l^T\mathbf{z} \le b_l \end{aligned} \tag{2.12}$$

Let us first consider (2.12):

- If $a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}\right) \le b_l$, then $\mathbf{z}_l^{(p)} = \mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}$ is the solution of (2.12). Substituting this $\mathbf{z}_l^{(p)}$ into (2.4), leads to the following update of $\mathbf{y}_l^{(p)}$:

$$\begin{aligned} & \mathbf{y}_l^{(p)} = \mathbf{y}_l^{(p-1)} + (1/\alpha)\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)} - \mathbf{x}^{(p-1)}\right) \\ \Rightarrow\ & \mathbf{y}_l^{(p)} = 0 \end{aligned} \tag{2.13}$$

- If $a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}\right) > b_l$, then the constraint is active. The optimization problem (2.12) aims to find the point in the half-space $a_l^T\mathbf{z} \le b_l$ that minimizes its distance to the point $\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}$ (which is outside that half-space). The solution is the projection of the point $\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}$ on the hyperplane $a_l^T\mathbf{z} = b_l$, which is given by the following formula:

$$\mathbf{z}_l^{(p)} = \mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)} - \frac{a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l\right) - b_l}{a_l^T a_l}a_l \tag{2.14}$$

Substituting this $\mathbf{z}_l^{(p)}$ into (2.4), leads to:

$$\begin{aligned} \mathbf{y}_l^{(p)} &= \mathbf{y}_l^{(p-1)} + \frac{1}{\alpha}\left(-\alpha\mathbf{y}_l^{(p-1)} - \frac{a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}\right) - b_l}{a_l^T a_l}a_l\right) \\ &= -\frac{a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}\right) - b_l}{\alpha a_l^T a_l}a_l \end{aligned} \tag{2.15}$$

Then defining $\gamma_l^{(p)} = a_l^T \left(\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}\right) - b_l$ yields

$$\mathbf{y}_l^{(p)} = -\frac{\gamma_l^{(p)}}{\alpha a_l^T a_l} a_l \tag{2.16}$$

If we define

$$\gamma_l^{(p)} = \max\{a_l^T \left(\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}\right) - b_l, 0\} \tag{2.17}$$

then we can use the update formula (2.16) for both cases.

Similarly, for the minimization under equality constraint (2.11), we define

$$\gamma_l^{(p)} = a_l^T \left(\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}\right) - b_l \tag{2.18}$$

and the update formula (2.16) gives the result of (2.4).

Now we consider step 4) of Algorithm 2.1. As shown in Boyd and Vandenberghe (2004), the function $q(\mathbf{x}) = \mathbf{x}^T H \mathbf{x}$ with H being a positive definite matrix, is uniformly convex on $\mathbb{R}^{n_\mathbf{x}}$ and has the conjugate function:

$$q^*(\mathbf{y}) = \frac{1}{2} \mathbf{y}^T H^{-1} \mathbf{y} \tag{2.19}$$

$$\Rightarrow \nabla q^*(\mathbf{y}) = H^{-1} \mathbf{y} \tag{2.20}$$

Consequently, in Han's algorithm for the definite quadratic program (2.1), it is not necessary to compute $\mathbf{z}^{(p)}$, and $\mathbf{y}^{(p)}$ can be eliminated using (2.16). We are now ready to describe the simplified Han's algorithm for problem (2.1), with the choice $\alpha = s/\rho$ (cf. footnote 2).

Algorithm 2.2 ***Han's algorithm for definite quadratic programs***

For each $l = 1, \ldots, s$, *compute*

$$c_l = \frac{-1}{\alpha a_l^T a_l} H^{-1} a_l \tag{2.21}$$

Define $\gamma_1^{(p)} \in \mathbb{R}, l = 1, \ldots, s$, *initialize* $\gamma_1^{(0)} = \cdots = \gamma_s^{(0)} = 0$ *and* $\mathbf{x}^{(0)} = 0$. *For* $p = 1, 2, \ldots$, *perform the following computations:*

1) *For each* l *corresponding to an equality constraint* ($l = 1, \ldots, n_{\text{eq}}$), *compute* $\gamma_l^{(p)} = a_l^T \mathbf{x}^{(p-1)} + \gamma_l^{(p-1)} - b_l$.
For each l *corresponding to an inequality constraint* ($l = n_{\text{eq}} + 1, \ldots, s$), *compute* $\gamma_l^{(p)} = \max\{a_l^T \mathbf{x}^{(p-1)} + \gamma_l^{(p-1)} - b_l, 0\}$;

2) Set

$$\mathbf{x}^{(p)} = \sum_{l=1}^{s} \gamma_l^{(p)} c_l \tag{2.22}$$

Note that Han's method splits up the computation into s parallel subproblems, where s is the number of constraints. However, although Algorithm 2.2 is simpler than the original form in Algorithm 2.1, it still requires a *global update scheme* and the parallel problems still operate with the full-sized decision vector. Implementing the scheme in a DMPC system, where the goal is to reduce the size of local computations and to rely on local communication between subsystems only, is not straightforward. In the following section, we will exploit the structure of the problem (2.1), resulting in a distributed algorithm that does not require global communications.

2.3 Distributed version of Han's method for the MPC problem

2.3.1 Distributed version of Han's method with common step size

The main idea behind the distributed version of Han's method is illustrated in Figure 2.1, with a simple example consisting of 4 subsystems and the coupling matrix that shows how subsystems are coupled via their variables (boxes on the same row indicate the variables that are coupled in one constraint). The figure represents the coupling pattern of this optimization problem:

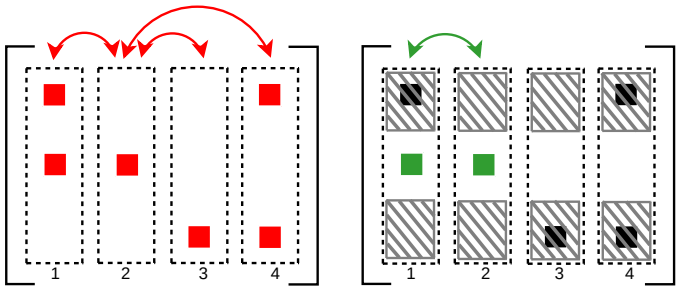
$$\begin{aligned} \min_{\mathbf{x}} \quad & x_1^2 + x_2^2 + x_3^2 + x_4^2 \\ \text{s.t.} \quad & x_1 + x_4 = 1 \\ & x_1 + x_2 \leq 2 \\ & x_3 + x_4 \leq 3 \end{aligned} \tag{2.23}$$

where each x_i is the variable of subsystem $i, i = 1, \ldots, 4$, and $\mathbf{x} = [x_1, x_2, x_3, x_4]^T$. The neighborhood sets are $\mathcal{N}^1 = \{1, 2, 4\}, \mathcal{N}^2 = \{2, 1\}, \mathcal{N}^3 = \{3, 4\}, \mathcal{N}^4 = \{4, 1, 3\}$.

In Han's parallel method, a subsystem has to communicate with all other subsystems in order to compute the updates of the global variable $\mathbf{x}^{(p)}$. For the distributed version of Han's method, each subsystem i only communicates with its neighboring subsystems for computing the updates of its local variables that involve variables of itself and its neighbors (these local variables will be defined in this section as *self-images* and *neighborhood images*). The definition of a subsystem's neighboorhood in this optimization problem is the same as $\mathcal{N}^i$ that is described in Section 1.2.1.

For the algorithm presented in this section, we use M local controllers attached to M subsystems. Each controller i then computes $\gamma_l^{(p)}$ with regards to a small set of constraints indexed by $l \in \mathfrak{L}_i$, where $\mathfrak{L}_i$ is a set of indices[4] of several constraints that involve subsystem i. Subsequently, it performs a local update for its own variables, such that the parallel local

[4]The choice of $\mathfrak{L}_i$ will be described on page 16.



(a) (b)

Figure 2.1: *Illustration of communication links of Han's algorithm for an example 4-subsystem problem with a sparsely coupled constraint matrix (each constraint is shown in a row), in which the first constraint involves variables of subsystems 1 and 4, the second constraint couples subsystems 1 and 2, and the third one couples subsystems 3 and 4. In the centralized coordination version (a), an update for a global variable requires subsystem 2 to communicate with all the others. In the distributed coordination version (b), the update of each row is done separately, therefore subsystem 2 only needs to communicate with the subsystem 1 to update its nonzero entry.*

update scheme will be equivalent to the global update scheme in Algorithm 2.2. We will also make use of the block-diagonal property of the Hessian matrix H.

Initialization of the algorithm

Store invariant parameters

The parameter α is chosen as in Algorithm 2.2 and stored in the memory of all local controllers.

We also compute s constant scalars c_l as in (2.21), in which each c_l corresponds to one constraint of (2.1). Note that since H is block-diagonal, H^{-1} can be computed easily by inverting each block of H and it has the same block structure as H. Hence c_l is as sparse as the corresponding a_l. We can see that c_l can be computed locally by a local controller with a priori knowledge of the parameter a_l and the weighting blocks on the diagonal of H that correspond to the non-zero elements of a_l.

Assign responsibility of each local controller

Each local controller is in charge of updating the variables of its subsystem. Moreover, we also assign to each local controller the responsibility of updating *some* intermediate variables that relate to several equality or inequality constraints in which its subsystem's states or inputs appear. The control designer has to assign each of the s scalar constraints to one of the M local controllers[5] such that the following requirements are satisfied:

[5]Note that s, the total number of constraints, is often much larger than M.

- Each constraint is taken care of by one and only one local controller (even for a coupled constraint, there will be only one controller that is responsible).
- A local controller can only be in charge of constraints that involve its own variables.

Let $\mathfrak{L}_i$ denote the set of indices l that local controller i is in charge of. The first requirement above can be written compactly as

$$\mathfrak{L}_i \cap \mathfrak{L}_j = \emptyset, \forall i, j \tag{2.24}$$

$$\mathfrak{L}_1 \cup \cdots \cup \mathfrak{L}_M = \{1, \ldots, s\} \tag{2.25}$$

and the second requirement can be mathematically expressed as

$$\text{If } l \in \mathfrak{L}_i \text{ then } a_l^i \neq 0 \tag{2.26}$$

where a_l^i is a subvector of a_l that corresponds to the variables of subsystem i in the vector $\mathbf{x}$.

Note that this partition is not unique and has to be created according to a procedure that is performed in the initialization phase. Recall that $\mathcal{N}^i$ denotes the neighborhood set of subsystem i, we also define $\mathfrak{L}_{\mathcal{N}^i}$ as the set of indices l corresponding to the constraints that are taken care of by subsystem i or by any neighbor of i:

$$\mathfrak{L}_{\mathcal{N}^i} = \bigcup_{j \in \mathcal{N}^i} \mathfrak{L}_j \tag{2.27}$$

If a local controller i is in charge of the constraints indexed by $l \in \mathfrak{L}_i$, then it computes locally c_l using (2.21) and exchanges these values with its neighbors. Then each local controller i stores $\{c_l\}_{l \in \mathfrak{L}_{\mathcal{N}^i}}$ in its memory throughout the optimization process.

Remark 2.4 *The partitioning of the constraint indices into the sets $\mathfrak{L}_i$ will affect the computation and communication loads assigned to local controllers, hence the partitioning should consider resources available at local controllers for a balancing purpose. However, the optimality property of the distributed algorithm does not depend on the partitioning of the constraints, as it will be shown later that its results are always equivalent to the results generated by the original centralized counterpart.*

Iterative procedure

The distributed algorithm consists of an iterative procedure running within each sampling interval. Since we want to obtain a feasible solution to the optimization problem (2.1) which could only be possible when Han's algorithm converges, we assume that the sampling time used is large enough such that the algorithm can converge within one sampling interval. This assumption will be used in Proposition 2.6, and its restrictiveness will be discussed in Section 2.5.

In the algorithm description, p is used to denote the iteration step. Values of variables obtained at iteration p are denoted with superscript (p).

Definition 2.1 (Index matrix of subsystems) *In order to present the algorithm compactly, we introduce the index matrix of subsystems: each subsystem i is assigned a square diagonal matrix $\mathfrak{I}^i \in \mathbb{R}^{n_{\mathbf{x}} \times n_{\mathbf{x}}}$, with an entry on the diagonal being 1 if it corresponds to the position of a variable of subsystem i in the vector $\mathbf{x}$, and 0 otherwise. In short, $\mathfrak{I}^i$ is a selection matrix such that the multiplication $\mathfrak{I}^i\mathbf{x}$ only retains the variables $u_0^i, \ldots, u_{N-1}^i, x_1^i, \ldots, x_N^i$ of subsystem i in its nonzero entries.*

From Definition 2.1 it follows that:

$$\sum_{i=1}^{M} \mathfrak{I}^i = I \tag{2.28}$$

Definition 2.2 (Self-image) *We denote with $\mathbf{x}^{(p)|i} \in \mathbb{R}^{n_{\mathbf{x}}}$ the vector that has the same size as $\mathbf{x}$, containing $u_0^{i,(p)}, \ldots, u_{N-1}^{i,(p)}, x_1^{i,(p)}, \ldots, x_N^{i,(p)}$ (i.e., the values of i's variables computed at iteration p) at the right positions, and zeros for the other entries. This vector is called the self-image of $\mathbf{x}^{(p)}$ made by subsystem i.*

Using the index matrix notation, the relation between $\mathbf{x}^{(p)|i}$ and $\mathbf{x}^{(p)}$ is:

$$\mathbf{x}^{(p)|i} = \mathfrak{I}^i \mathbf{x}^{(p)} \tag{2.29}$$

Definition 2.3 (Neighborhood image) *Extending the concept of self-image, we denote with $\mathbf{x}^{(p)|\mathcal{N}^i}$ the neighborhood image of subsystem i made from $\mathbf{x}$. At step p of the iteration, subsystem i constructs $\mathbf{x}^{(p)|\mathcal{N}^i}$ by putting the values of its neighbors' variables and its own variables into the right positions, and filling in zeros for the remaining slots of $\mathbf{x}$. The neighborhood image $\mathbf{x}^{(p)|\mathcal{N}^i}$ satisfies the following relations:*

$$\mathbf{x}^{(p)|\mathcal{N}^i} = \sum_{j \in \mathcal{N}^i} \mathbf{x}^{(p)|j} \tag{2.30}$$

$$\mathbf{x}^{(p)|\mathcal{N}^i} = \left(\sum_{j \in \mathcal{N}^i} \mathfrak{I}^j \right) \mathbf{x}^{(p)} \tag{2.31}$$

By definition, we also have the following relation between the self-image and the neighborhood image made by the same subsystem:

$$\mathbf{x}^{(p)|i} = \mathfrak{I}^i \mathbf{x}^{(p)|\mathcal{N}^i} \tag{2.32}$$

Using the notation described above, we now describe the subtasks that each controller will use in the distributed algorithm.

- **Communications with the neighbors**

  Each controller i communicates only with its neighbors $j \in \mathcal{N}^i$ to get updated values of their variables and sends its updated variables to them. The data that each subsystem i transmits to its neighbor $j \in \mathcal{N}^i$ consists of the self-image $\mathbf{x}^{(p)|i}$ and

the intermediate variables $\gamma_l^{(p)}, l \in \mathfrak{L}_i$, which are maintained locally by subsystem i.

- **Update dual variables** γ_l

  The local controller i updates γ_l corresponding to each constraint $l \in \mathfrak{L}_i$ under its responsibility in the following manner:

  – If constraint l is an equality constraint ($l \in \{1, \ldots, n_{\text{eq}}\}$), then

  $$\gamma_l^{(p)} = a_l^T \mathbf{x}^{(p-1)|\mathcal{N}^i} + \gamma_l^{(p-1)} - b_l \tag{2.33}$$

  – If constraint l is an inequality constraint ($l \in \{n_{\text{eq}} + 1, \ldots, s\}$), then

  $$\gamma_l^{(p)} = \max\{a_l^T \mathbf{x}^{(p-1)|\mathcal{N}^i} + \gamma_l^{(p-1)} - b_l, 0\} \tag{2.34}$$

- **Update primal variables**

  Local controller i uses all $\gamma_l^{(p)}$ values that it has (by communications and those computed by itself) to compute an 'assumed neighborhood image' $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$. Note that $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ has the same structure as the neighborhood image $\mathbf{x}^{(p)|\mathcal{N}^i}$. However, in $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$, only the values of variables of subsystem i are guaranteed to be correct, while the other values can be outdated, hence $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ is not the exact update of the neighborhood image. Indeed, $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ is used only for constructing the new self-image by selecting the variables $u_0^{i,(p)}, \ldots, u_{N-1}^{i,(p)}, x_1^{i,(p)}, \ldots, x_N^{i,(p)}$ of subsystem i in $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$:

  $$\mathbf{x}^{(p)|i} = \mathfrak{I}^i \mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i} \tag{2.35}$$

- **Check the local termination criteria**

  For each local controller, there are local termination criteria. Each controller checks the local termination criteria using local communications only[6]. The local termination criteria also aim to keep a subsystem informed when other subsystems are ready to terminate. When a subsystem's local termination criteria are satisfied and it is informed that the other subsystems' termination criteria are also satisfied, the algorithm stops and the local control actions are implemented.

In the following, we will present the distributed algorithm.

Algorithm 2.3 ***Distributed algorithm for positive definite quadratic programs***

Initialize with $p = 0$, $u_k^{i,(0)} = 0, x_{k+1}^{i,(0)} = 0, \forall i, k = 0, \ldots, N-1$ *(this means* $\mathbf{x}^{(0)|i} = 0, \forall i$*, implying* $\mathbf{x}^{(0)} = 0$*), and* $\gamma_l^{(0)} = 0, l = 1, \ldots, s$

[6]Checking the termination criteria in a distributed fashion requires a dedicated logic scheme; several schemes were described in (Bertsekas and Tsitsiklis, 1989, Chapter 8).

Next, for $p = 1, 2, \ldots$, the following steps are executed in parallel and with synchronization:

1) ***Communications to get the updated primal variables***

   Each controller i gets updated values of $\mathbf{x}^{(p-1)|j}$ from its neighbors $j \in \mathcal{N}^i$, where only non-zero elements need to be transmitted[7].

   Then controller i constructs the neighborhood image $\mathbf{x}^{(p-1)|\mathcal{N}^i}$ *using formula* (2.30).

2) ***Update dual variables γ_l in parallel***

   Each local controller i updates γ_l for each $l \in \mathfrak{L}_i$, using (2.33) *or* (2.34).

3) ***Communications to get the updated intermediate variables***

   Each local controller i gets $\gamma_l^{(p)}, l \in \mathfrak{L}_{\mathcal{N}^i}$ from its neighbors $j \in \mathcal{N}^i$.

4) ***Update primal variables in parallel***

   Each local controller i computes an assumed neighborhood image *of* $\mathbf{x}$*:*

$$\mathbf{x}_{assumed}^{(p)|\mathcal{N}^i} = \sum_{l \in \mathfrak{L}_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \tag{2.36}$$

   Then controller i constructs the new self-image, *using* (2.35).

5) ***Check the local termination criteria in parallel***

   Each local controller checks the local termination criteria. If local termination criteria are satisfied, the algorithm stops, otherwise go to step 1) to start a new iteration.

In Algorithm 2.3, the activities of one local controller can be demonstrated by the diagram in Figure 2.2. The diagram clearly shows that in the distributed algorithm, each local controller i only communicates with its neighbors $j \in \mathcal{N}^i$, enabling implementation of the method in a distributed setting. The properties of the distributed algorithm will be discussed in the following subsections.

Proof of equivalence to Han's algorithm using a global update scheme

In Algorithm 2.2, at step 2), the centralized variable $\mathbf{x}^{(p)}$ is updated via a global update scheme. In Algorithm 2.3, by the local update scheme we obtain $\mathbf{x}^{(p)|i}$ for $i = 1, \ldots, M$. The equivalence of these two algorithms is stated in the following proposition:

Proposition 2.4 *Applying Algorithms 2.2 and 2.3 to the same problem* (2.1) *with the same parameter α, at any iteration p, the following statements hold:*

[7] Since $\mathbf{x}^{(p-1)|i}$ only has a few non-zero elements, which are $u_0^{i,(p-1)}, \ldots, u_{N-1}^{i,(p-1)}$, $x_1^{i,(p-1)}, \ldots, x_N^{i,(p-1)}$, only these values need to be transmitted by controller i to reduce communications.

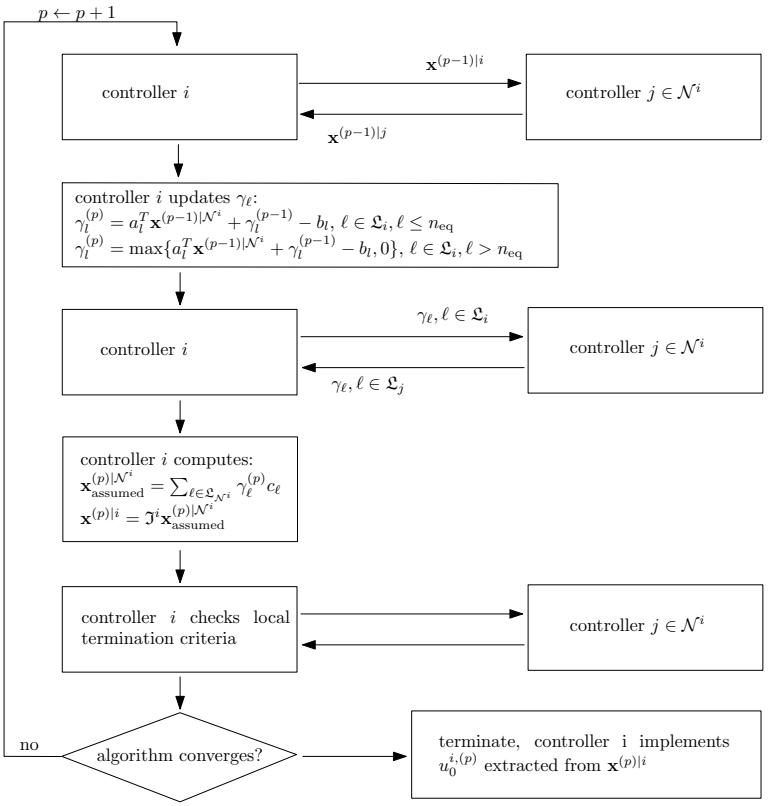


Figure 2.2: *Computation and communication flow-chart of controller i in each iteration of Algorithm 2.3. Controller i only needs to communicate with its neighbors $j \in \mathcal{N}^i$.*

a) $\gamma_l^{(p)}$ *are the same in Algorithms 2.2 and 2.3, for all* $l \in \{1, \ldots, s\}$.

b) $\mathbf{x}^{(p)} = \sum_{i=1}^{M} \mathbf{x}^{(p)|i}$, *in which* $\mathbf{x}^{(p)}$ *is calculated in Algorithm 2.2 while* $\mathbf{x}^{(p)|i}, i = 1, \ldots, M$ *are calculated in Algorithm 2.3.*

Hence, Algorithm 2.2 and Algorithm 2.3 generate the same solution.

Proof: The proposition will be proved by induction.

It is clear that statements a) and b) hold for $p = 0$.

Now consider iteration p, and assume that the statement a) and b) hold for all iterations before iteration p.

First, we prove statement a). For any l and i such that $l \in \mathfrak{L}_i$, we have:

$$a_l^T \mathbf{x}^{(p-1)} = a_l^T \left(\sum_{j=1}^{M} \mathfrak{I}^j \right) \mathbf{x}^{(p-1)} \tag{2.37}$$

$$= a_l^T \sum_{j=1}^{M} \mathbf{x}^{(p-1)|j} \tag{2.38}$$

$$= a_l^T \sum_{j=1}^{M} \mathfrak{J}^j \mathbf{x}^{(p-1)|j} \tag{2.39}$$

$$= a_l^T \left(\sum_{j \in \mathcal{N}^i} \mathfrak{J}^j \mathbf{x}^{(p-1)|j} + \sum_{j \notin \mathcal{N}^i} \mathfrak{J}^j \mathbf{x}^{(p-1)|j} \right)$$

Due to the definition of *neighborhood*, a subsystem outside $\mathcal{N}^i$ does not have any coupled constraints with subsystem i. Therefore, $a_l^T \sum_{j \notin \mathcal{N}^i} \mathfrak{J}^j \mathbf{x}^{(p-1)|j} = 0$, which - in combination with (2.30) - leads to:

$$a_l^T \mathbf{x}^{(p-1)} = a_l^T \sum_{j \in \mathcal{N}^i} \mathfrak{J}^j \mathbf{x}^{(p-1)|j} = a_l^T \mathbf{x}^{(p-1)|\mathcal{N}^i} \tag{2.40}$$

Equation (2.40) then guarantees that $\gamma_l^{(p)}$ computed at step 1) of Algorithm 2.2 and at step 2) of Algorithm 2.3 are the same.

Now consider statement b), where the main argument is the following: The same set of $\gamma_l^{(p)}$ and c_l are used for updating i's variables in $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ (at step 4 of Algorithm 2.3) and in $\mathbf{x}^{(p)}$ (at step 2 of Algorithm 2.2). Thus each vector of local update $\mathbf{x}^{(p)|i}$, which contains values of i's variables selected from $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$, is a part of the centralized update $\mathbf{x}^{(p)}$.

More specifically, we can express the formula of $\mathbf{x}^{(p)|i}$ computed in Algorithm 2.3 as

$$\begin{aligned} \mathbf{x}^{(p)|i} = \mathfrak{J}^i \mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i} &= \mathfrak{J}^i \sum_{l \in \mathfrak{L}_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \\ \Rightarrow \sum_{i=1}^{M} \mathbf{x}^{(p)|i} &= \sum_{i=1}^{M} \mathfrak{J}^i \sum_{l \in \mathfrak{L}_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \end{aligned} \tag{2.41}$$

Note that in the following equations, $\mathbf{x}^{(p)}$ refers to the update of the decision variable computed by (2.22) in Algorithm 2.2, which we can express as

$$\mathbf{x}^{(p)} = \sum_{i=1}^{M} \mathfrak{J}^i \mathbf{x}^{(p)} = \sum_{i=1}^{M} \mathfrak{J}^i \sum_{l=1}^{s} \gamma_l^{(p)} c_l \tag{2.42}$$

in which the first equality is due to the relation (2.28), the second equality is from (2.22).

Recall that c_l has the same structure as a_l, and if $l \notin \mathfrak{L}_{\mathcal{N}^i}$ then a_l and c_l do not have any non-zero values at the positions associated with variables of subsystem i. Therefore

$$\mathfrak{J}^i \sum_{l=1}^{s} \gamma_l^{(p)} c_l = \mathfrak{J}^i \left(\sum_{l \notin \mathfrak{L}_{\mathcal{N}^i}} \gamma_l^{(p)} c_l + \sum_{l \in \mathfrak{L}_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \right) = \mathfrak{J}^i \sum_{l \in \mathfrak{L}_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \tag{2.43}$$

This equality shows that (2.42) and (2.41) are equivalent, thus proving the equality in statement b): $\mathbf{x}^{(p)} = \sum_{i=1}^{M} \mathbf{x}^{(p)|i}$. □

The equivalent result of Algorithms 2.2 and 2.3 implies that problem (2.1) can be solved using Algorithm 2.3. This allows us to implement a DMPC scheme using Algorithm 2.3 that does not need global communications and global computations, each local controller only exchanges the updates of its predicted variables with its direct neighbors.

2.3.2 Properties of the distributed model predictive controller based on Han's method

Convergence, feasibility, and stability properties of the DMPC scheme using Algorithm 2.3 are established by the following propositions:

Proposition 2.5 *Assume that problem* $(\mathcal{P})$ *in* (1.10)–(1.14) *has a feasible solution. Then Algorithm 2.3 asymptotically converges to the centralized solution of* $(\mathcal{P})$ *at each sampling step.*

Proof: In Han and Lou (1988) it is shown that Han's method is guaranteed to converge to the centralized solution of the convex quadratic program (2.1) under the conditions that $q(\mathbf{x})$ is uniformly convex with the coefficient ρ and differentiable on $\mathbb{R}^{n\mathbf{x}}$, (2.1) has a feasible solution, and the step size α satisfies $\alpha \leq s/\rho$. Due to the positive definiteness of Q_i and R_i, and the assumption that $(\mathcal{P})$ has a feasible solution, such conditions hold for the quadratic problem (2.1). Moreover, Algorithm 2.3 is equivalent to Han's method for the problem (2.1). Hence, the distributed scheme in Algorithm 2.3 converges to the centralized solution of (2.1), which is the same as $(\mathcal{P})$. □

Proposition 2.6 *Assume that at every sampling step, Algorithm 2.3 asymptotically converges. Then the DMPC scheme is recursively feasible and stable.*

Proof: By letting Algorithm 2.3 converge at every sampling step, the centralized solution of $(\mathcal{P})$ is obtained. Recursive feasibility and stability is guaranteed as a consequence of centralized MPC with a terminal point constraint, as shown in Mayne et al. (2000) and Keerthi and Gilbert (1988). □

2.3.3 Distributed version of Han's method with scaled step size

A disadvantage of Han's method (and its distributed version) is the slow convergence rate, due to the fact that it is essentially a projection method to solve the dual problem of (2.1). Moreover, in order to find the initial guesses of the primal and the conjugate variables that have to satisfy the relation (2.5) with $p = 0$, Han's (distributed) method uses zeros as the initial guesses, which prevents warm starting of the algorithm by choosing an initial guess that is close to the optimizer. Therefore, we need to modify the method to achieve a better convergence rate.

In this section, we present two modifications of the distributed version of Han's method:

- Scaling of the step sizes related to dual variables by using heterogeneous α_l for the update of the l-th dual variable instead of the same α for all dual variables. Moreover there is a new parameter for increasing the step size, in order to accelerate the convergence.

- Use of nonzero initial guesses, which allows taking the current MPC solution as the start for the next sample step.

Note that Han's method can be extended to the case using heterogeneous α_l for the l-th dual variable, this extended version of Han's method can also be proved for convergence with we use $\alpha_l \geq s/\rho,\ \forall l$. Indeed, following the same line of proof of Han's method (see the summary in Remark 2.2), we can see α_l will replace α in (2.10) for each residual of the l-th dual variable, and hence the convergence is guaranteed if $\alpha_l\rho - s \geq 0,\ \forall l$, or $\alpha_l \geq s/\rho,\ \forall l$. However, this condition still leads to a slow convergence. The major factor to accelerate the convergence is to increase the step sizes, i.e., to use α_l below the guaranteed value. Thus, the convergence of the modified distributed algorithm is not guaranteed; this will be discussed in Section 2.5.

In order to implement the above modifications, the improved distributed version of Han's method is initialized similarly to the distributed algorithm in Section 2.3.1, except for the following procedures:

1. **Pre-computed invariant parameters**

   Each subsystem i computes and stores the following parameters throughout the control scheme:

   - For each $l \in \mathfrak{L}_i$: $\alpha_l = \left(k_\alpha\right)_l \alpha_0$, where k_α is the scaling vector. The scalar α_l acts as local step size regarding the l-th dual variable, and therefore k_α should be chosen such that the convergence rates of all s dual variables are improved. The method to choose k_α will be discussed in Remark 2.5.
   - For each $l \in \mathfrak{L}_i$: $\bar{c}_l = \frac{-1}{a_l^T a_l} H^{-1} a_l$. We can see that $\bar{c}_l$ can be computed locally by a local controller with a priori knowledge of the parameter a_l and the weighting blocks on the diagonal of H that correspond to the non-zero elements of a_l.

2. **MPC step**

   At the beginning of the MPC step, the current states of all subsystems are measured. The sequences of predicted states and inputs generated in the previous MPC step are shifted forward one step, then we append zero states and zero inputs to the shifted sequences. The new sequences are then used as the initial guess for solving the optimization problem in the current MPC step[8]. The initial guess for each subsystem can be defined locally. For subsystem i, denote the initial guess as $\mathbf{x}^{(0)|i}$. At the first MPC step, we have $\mathbf{x}^{(0)|i} = 0, \forall i$.

[8]The idea of using previously predicted states and inputs for initialization is a popular technique in MPC (Rawlings and Mayne, 2009). Especially with Han's method, whose convergence rate is slow, an initial guess that is close to the optimal solution will be very helpful to reduce the number of iterations.

The current state is plugged into the MPC problem, and then we get an optimization problem of the form (2.1). This problem will be solved by the following modified distributed algorithm of Han's method.

Algorithm 2.7 ***Improved distributed algorithm for the MPC optimization problem***

Initialize with $p = 0$. Each subsystem i uses the initial guess as $\mathbf{x}^{(0)|i}$.

Next, for $p = 1, 2, \ldots$, the following steps are executed in parallel and with synchronization:

1) *See step 1 of Algorithm 2.3.*

2) *See step 2 of Algorithm 2.3, except that for $p = 1$, each subsystem i computes the initial intermediate variables by*[9]*:*

$$\gamma_l^{(1)} = a_l^T \left(\mathbf{x}^{(0)|\mathcal{N}^i} - \frac{\alpha_l}{s} H \mathbf{x}^{(0)|\mathcal{N}^i} \right) - b_l, \quad l \in \mathfrak{L}_i, l \leq n_{\text{eq}} \tag{2.44}$$

$$\gamma_l^{(1)} = \max \left\{ a_l^T \left(\mathbf{x}^{(0)|\mathcal{N}^i} - \frac{\alpha_l}{s} H \mathbf{x}^{(0)|\mathcal{N}^i} \right) - b_l, 0 \right\}, \quad l \in \mathfrak{L}_i, l > n_{\text{eq}} \tag{2.45}$$

3) *See step 3 of Algorithm 2.3.*

4) *See step 4 of Algorithm 2.3 but with a different formula to update the assumed neighborhood image for each i:*

$$\mathbf{x}_{assumed}^{(p)|\mathcal{N}^i} = \sum_{l \in \mathfrak{L}_{\mathcal{N}^i}} \frac{1}{\alpha_l} \gamma_l^{(p)} \bar{c}_l \tag{2.46}$$

5) *See step 5 of Algorithm 2.3.*

When the iterative procedure finishes, each subsystem applies the first input $u_0^{i,(p)}$, then waits for the next state measurement to start a new MPC step.

Remark 2.5 *The main improvement of Algorithm 2.7 over Algorithm 2.3 is the improved convergence speed, which heavily depends on a good choice of the scaling vector k_α. We have observed that the convergence speed of some dual variables under the responsibility of a subsystem i will affect the convergence speed of dual variables under the responsibility of its neighbors in $\mathcal{N}^i$. Therefore the choice of scaling vector should focus on improving the convergence speed of dual variables that appear to converge more slowly. In our case, we rely on the Hessian to find the scaling vector. Specifically, for each subsystem i, let $\bar{h}_i$*

[9] The intermediate variables are constructed following the formulas (2.17)–(2.18) with replacing the common α by α_l for each $l \in \{1, \ldots, s\}$, where we implicitly use $\mathbf{y}_l^{(0)} = \frac{1}{s}\mathbf{y}^{(0)}, \forall l \in \{1, \ldots, s\}$ and $\mathbf{y}^{(0)} = H\mathbf{x}^{(0)}$. Also note that since a_l only involves neighboring subsystems and H is block-diagonal, the computation $a_l^T \left(\mathbf{x}^{(0)} - \frac{\alpha_l}{s} H \mathbf{x}^{(0)} \right)$ only uses values from neighboring subsystems, similarly to the argument for (2.40).

denote the average weight of its variables (i.e., average of entries related to i's states and inputs in the diagonal of the Hessian). We then choose the scaling factor $\left(k_\alpha\right)_l = 1/\bar{h}_i$, for all $l \in \mathfrak{L}_i$. We also multiply the scaling vector k_α with a factor $\theta \in (0,1)$ for enlarging the step sizes of all dual variables. In the first MPC step, we do the tuning offline with $\theta \simeq 1$ and gradually reduce θ until it causes the algorithm to diverge, then we stop and choose the smallest θ such that the algorithm still converges.

The choice of the scaling vector depends on the structure of the centralized optimization problem, thus we only need to choose it once in the first MPC step. Then for the next MPC steps, we can re-use the same scaling vector.

The efficiency of Algorithm 2.7 will be demonstrated in the example of irrigation canal control, which is presented in the next section.

2.4 Application of Han's method for distributed MPC in canal systems

2.4.1 The example canal system

The novel DMPC approach is applicable to a wide range of large-scale systems that could be modeled in the LTI form as described in Section 1.2. In this section, we demonstrate its application in an example control problem, where the objective is to regulate the water flows in a system of irrigation canals. Irrigation canals are large-scale systems, consisting of many interacting components, and spanning vast geographical areas. For the most efficient and safe operation of these canals, maintaining the levels of the water flows close to pre-specified reference values is crucial, both under normal operating conditions as well as in extreme situations. Manipulation of the water flows in irrigation canals is typically done using devices such as pumps and gates.

The example irrigation canal to be considered is a 4-reach canal system as illustrated in Figure 2.3. In this system, water flows from an upstream reservoir through the reaches, under the control of 4 gates and a pump at the end of the canal system that discharges water.

The control design is based on the master-slave control paradigm, in which the master controllers compute the flows through the gates, while each slave controller uses the local control actuators to guarantee the flow set by the master controller (Schuurmans et al., 1999). We will use the new DMPC method to design the master controllers.

2.4.2 Modeling the canal

The canal system is divided into 4 subsystems, each of which corresponds to a reach and also includes the local controller at the upstream gate of the reach. The 4^{th} subsystem has one more controller, corresponding to the pump at its downstream end.

We use a simplified model for each subsystem as illustrated in Figure 2.4, and then obtain the overall model by connecting the subsystem models. A subsystem is approximately modeled by a reservoir with upstream in-flow and downstream out-flow.

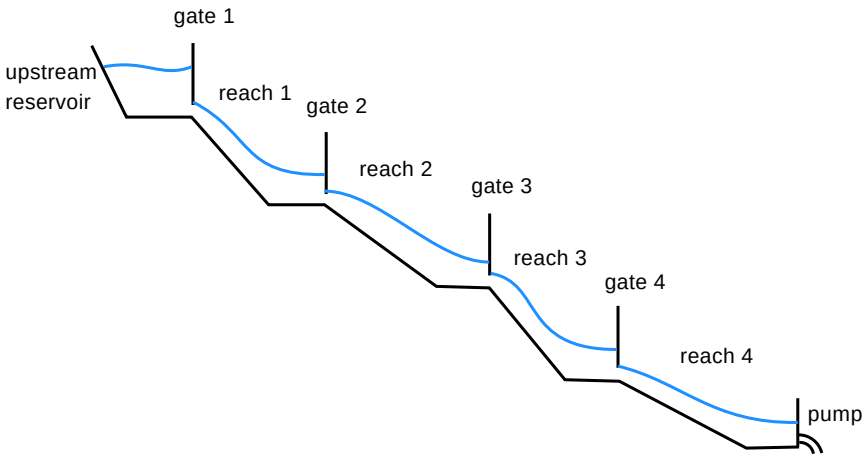


Figure 2.3: *The example canal system*

The discrete-time model of reach i is represented by:

$$h_{k+1}^i - h_k^i = \frac{T_\mathrm{s}}{A_\mathrm{s}^i}\left[\left(Q_\mathrm{in}^i\right)_k - \left(Q_\mathrm{out}^i\right)_k\right] \tag{2.47}$$

where superscript i represents the subsystem index, subscript k is for the time index, T_s is the sampling time, h is the downstream water level of the reach, A_s is the water surface (i.e., the volume of reservoir equals $h \cdot A_\mathrm{s}$), Q_in and Q_out are the in-flow and the out-flow of the canal which are measured at the upstream and downstream ends, respectively. Denote the flow passing the i^th gate by q^i, and the flow passing the pump by p^4. Due to the mass conservation law, we have $\left(Q_\mathrm{out}^i\right)_k = \left(Q_\mathrm{in}^{i+1}\right)_k = q_k^{i+1}$, for $i = 1, 2, 3$, and $\left(Q_\mathrm{out}^4\right)_k = p_k^4$.

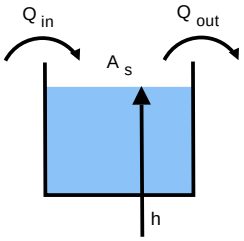


Figure 2.4: *Model of a reach*

In order to derive local dynamics, we choose state and input vectors of subsystem i as

$$x_k^i = h_k^i$$
$$u_k^i = \begin{cases} q_k^i & , \quad i = 1, 2, 3 \\ \begin{bmatrix} q_k^i \\ p_k^i \end{bmatrix} & , \quad i = 4 \end{cases}$$

The dynamics of each subsystem can be represented by a discrete-time, linear time-invariant model of the form (1.1) with the state-space matrices:

$$
\begin{aligned}
&A^{ii} = \ 1 \ , \quad i = 1, \ldots, 4; \quad A^{ij} = \ 0 \ , \quad i \neq j \\
&B^{ii} = \ T_{\mathrm{s}}/A_{\mathrm{s}}^{i} \ , \quad i = 1, 2, 3; \quad B^{44} = \begin{bmatrix} T_{\mathrm{s}}/A_{\mathrm{s}}^{4} & -T_{\mathrm{s}}/A_{\mathrm{s}}^{4} \end{bmatrix} \\
&B^{i(i+1)} = \ -T_{\mathrm{s}}/A_{\mathrm{s}}^{i} \ , \quad i = 1, 2; \quad B^{34} = \begin{bmatrix} -T_{\mathrm{s}}/A_{\mathrm{s}}^{4} & 0 \end{bmatrix} \\
&B^{ij} = 0 \ , \quad i = 1, 2, 3, \quad j \notin \{i, i+1\}.
\end{aligned}
$$

The control problem also includes physical constraints on all the water levels and flows. Let us denote the aggregate state and control input vectors at time step k as x_k and u_k, respectively. The physical constraints are the upper and the lower bounds of states and inputs as:

$$
\begin{aligned}
x_{\min} \leq x_k \leq x_{\max}, \ \forall k \\
u_{\min} \leq u_k \leq u_{\max}, \ \forall k
\end{aligned}
$$

In the simulation, after normalizing the parameters such that $B^{ii} \simeq 1, \ i = 1, 2, 3$, we set values of the bounds as $x_{\min} = -0.5 I_4, x_{\max} = 0.5 I_4$, and also $u_{\min} = -0.5 I_5, u_{\max} = 0.5 I_5$.

The control objective is to regulate the water levels in all the reaches when the initial water levels are deviated from the normal conditions. Note that the dynamical system is marginally stable and the B matrix has a block-upper triangular structure, this helps to easily design a controller to accomplish this objective and result in a stable closed-loop system. Therefore, the simulation is focused on the convergence property of the optimization algorithms, not on the stability of the MPC closed-loop system.

2.4.3 Simulation results

The DMPC methods developed in Section 2.3 are applied to the regulation problem of the simulated canal system described in previous subsections using the sampling time $T_s = 240s$, the prediction horizon $N = 10$, with a nonzero initial state. We use the distributed Han's method with and without the modifications described in Section 2.3, and compare the results. Figure 2.5 shows the convergence of the distributed solutions to the centralized solution for the problem. Starting from the same initial guess in the first MPC step, i.e., all variables are initialized with zeros, the distributed algorithm with modifications achieves a better convergence rate, allowing the distributed optimization to converge within an acceptable number of iterations. Similar results were also achieved for the next MPC steps, when we simulate the closed-loop MPC with Algorithm 2.7 and let the distributed solutions converge to the centralized solution at every step, with maximally 100 iterations per step. A simulation of closed-loop MPC is performed for 20 sampling steps. Figure 2.6 confirms that that the distributed solutions converge to the centralized solutions within 100 iterations per sampling step. Hence the solutions of the distributed MPC algorithm are also the globally optimal solutions.

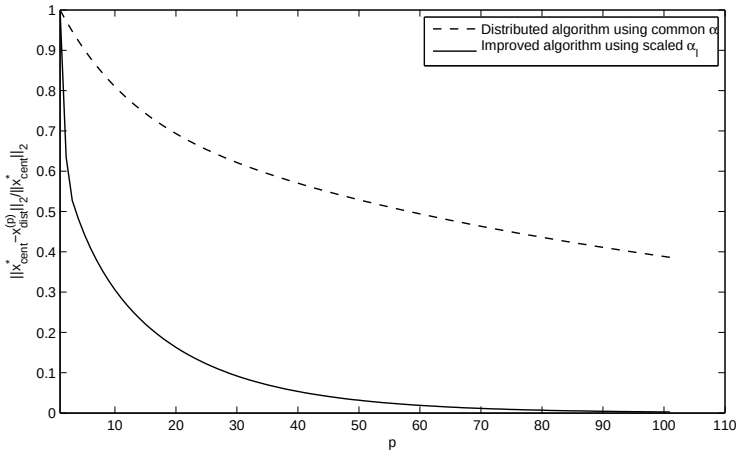


Figure 2.5: *Comparison of convergence speeds of two distributed versions of Han's method for the first sampling time step (k=1)*

2.5 Discussion on distributed Han's methods

Two distributed versions of Han's method have been described in Section 2.3, followed by a short demonstration of their usage in Section 2.4. Although these algorithms help to implement Han's method in a distributed setting for MPC, there are still some theoretical issues that need to be addressed.

Firstly, the proposed distributed algorithms deal with positive definite quadratic programs only. Although many MPC problems for linear time-invariant systems are formulated as quadratic programs, there are other variants that use different objective functions, and nonlinear MPC would also yield more complicated optimization problems than quadratic programs. With such problems, we might not be able to implement Han's parallel method in a distributed fashion. This issue motivates the research on other decomposition methods that can handle more general problems, e.g., convex problems with linear or decoupled nonlinear constraints. Although Han's parallel algorithm can still be applied to general nonlinear convex programs, it is difficult to derive a distributed version of Han's method for most of the nonlinear problems.

It is worth to address the conservativeness of the MPC formulation using the terminal point constraint $x_N = 0$, which reduces the domain of attraction of MPC. However, this issue is not related to Han's method. In case we want to use less conservative MPC, e.g., MPC with a terminal constraint set and a terminal controller, we need to find a separable terminal penalty function and local terminal constraint sets. However, to the authors' best knowledge, there is still no distributed scheme available to construct local terminal constraint sets and local terminal controllers (and also the terminal penalty matrix that is solution of the Riccati equation), other than assuming them to be completely decoupled. Therefore, although distributed Han's method can also be applied to any uniformly convex QP problem with a sparse coupling structure, it requires further research on MPC formu-

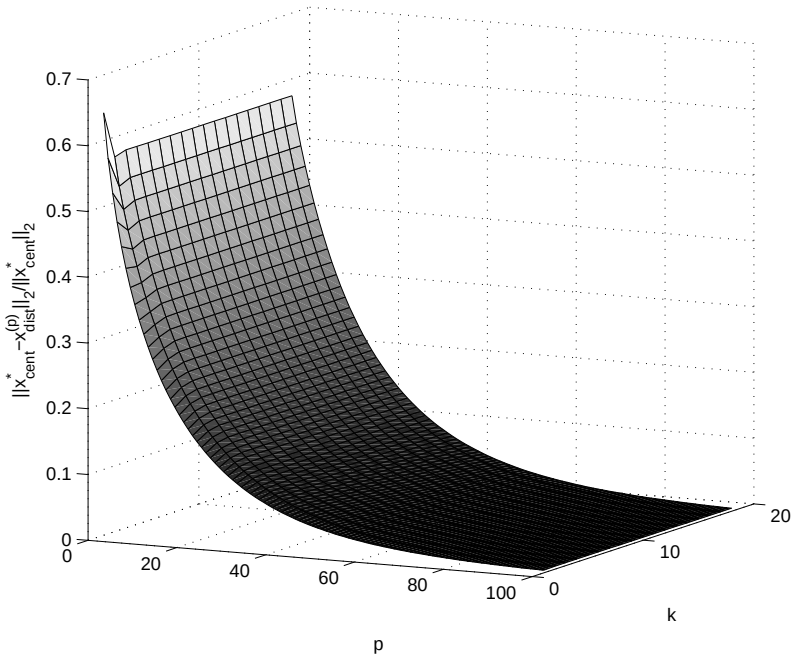


Figure 2.6: *Normalized norm of the difference between the centralized and the distributed solutions using Algorithm 2.7 versus iteration step p and sampling time step k*

lations that have such optimization problems and can guarantee stability with centralized MPC.

In general, Han's method has a slow convergence rate due to its iterative projection nature, which is inherited by Algorithm 2.3. Since the feasibility and stability properties are derived upon convergence of the algorithm within each sampling step, we need to speed up the convergence of this method. The distributed version of Han's method with scaling can improve the convergence rate significantly, as was illustrated in Section 2.4. However, its proof of convergence is still lacking. We observe that in setups that are more complicated, the proposed method to choose the scaling vector does not always work well (sometimes after several sample steps, the algorithm does not converge anymore). Due to the requirement not to have global communications, it is difficult to adjust the scaling vector during the iteration to reach convergence. Therefore, speeding up Han's method while providing a proof for convergence is still an open issue, and we may use a coordinator at a higher level of hierarchy that has global communication capabilities to tackle this issue.

Another issue is due to the formulation of the optimization problem for MPC, where we keep both inputs and states as variables of the centralized optimization problem and do not eliminate the states using the dynamic model equations. This formulation is advantageous in distributed MPC because the Hessian will then keep a block diagonal structure, and the neighborhood of each subsystem will only contain its direct neighbors (the neighborhood

would be greatly extended if we eliminate the states in the optimization problem). However, using states as variables requires considering the dynamical equations as equality constraints of the optimization problem, and the existence of equality constraints typically requires an exact solution in order to guarantee feasibility. Since Han's method converges asymptotically, we may not be able to get the exact optimal multipliers in real-time, and then the corresponding primal iterates would not be guaranteed to be feasible. In general, most dual decomposition methods do not provide primal feasible solutions before reaching the dual optimal solutions, so this feasibility issue also applies to other dual decomposition methods.

2.6 Conclusions

A decomposition approach based on Fenchel's duality and Han's parallel method has been developed in this chapter, resulting in two distributed algorithms that are applicable to DMPC. The first distributed algorithm generates updates that are the same as those computed globally by Han's method for definite quadratic problems, and therefore it has the same convergence property as Han's method. Moreover, feasibility and stability of DMPC are achieved upon convergence of the iterations. The second distributed algorithm aims to improve the convergence speed by using scaled step sizes and nonzero initial guesses. The new methods have been applied to an example irrigation canal network, demonstrating their applicability for water networks and can also be applied to other networked systems. We have also summarized open issues of using Han's method and other dual decomposition methods for MPC, including the topics of separable problem formulation, convergence rate, primal feasibility, and stability of MPC. Several issues will be addressed in the next chapters.

Chapter 3

Distributed proximal gradient methods for convex optimization problems

We consider optimization problems with mixed L_1/L_2-norm terms in the cost function and sparsely coupled constraints, and propose distributed optimization algorithms based on classical proximal gradient and accelerated proximal gradient methods using dual decomposition. We show that the distributed algorithm based on classical proximal gradient method is similar to distributed Han's algorithm described in Chapter 2 and has the low convergence rate for first-order gradient methods, $O(\frac{1}{k})$, where k is the iteration number. The distributed algorithm based on accelerated proximal gradient method achieves a significantly higher convergence rate, $O(\frac{1}{k^2})$. We also provide the optimal step-size of the presented algorithms.

3.1 Introduction

Han's method and its distributed versions presented in Chapter 2 can be classified as optimization algorithms that use first-order derivatives of the cost function; from now on we refer to this class of algorithms as *gradient-based methods*. The fact that Han's method exhibits slow convergence is not surprising, as it is well-known that classical gradient-based methods often have poor convergence rates. It has been shown in Bertsekas (1999), Nesterov (2004) that for functions with a Lipschitz-continuous gradient, i.e., smooth functions, classical gradient-based methods converge with the function error reduced at a rate of $O(\frac{1}{k})$, where k is the iteration number. Nemirovskii and Yudin (1983) have shown that a lower bound on the convergence rate for gradient-based methods is $O(\frac{1}{k^2})$. Nesterov has shown in his work (Nesterov, 1983) that an accelerated gradient algorithm can be constructed such that this lower bound on the convergence rate is achieved when minimizing unconstrained smooth functions. This result has been extended and generalized in several publications to handle constrained smooth problems and smooth problems with an additional non-smooth term (Nesterov, 1988, 2005, Beck and Teboulle, 2009, Tseng, 2008).

Gradient-based optimization methods are known for their simplicity and low complexity within each iteration. Hence they are favorable for distributed implementation, especially when they are used in combination with dual decomposition techniques. Recently, distributed MPC approaches based on distributed gradient-based methods for the dual problem have been studied (Giselsson and Rantzer, 2010, Negenborn et al., 2008), as well as the distributed version of Han's algorithm presented in Chapter 2. The above mentioned methods rely on gradient-based optimization, which suffers from low convergence rates. In addition, the step size parameter in the gradient scheme must be chosen appropriately to get a good performance. Such information has not been provided or has been chosen conservatively in these distributed MPC approaches.

In this chapter we propose a new distributed optimization algorithm that can be used for distributed MPC. The underlying algorithm is an accelerated gradient method to solve the dual problem instead of the classical gradient method. We also extend the class of problems considered by allowing an additional sparse but non-separable 1-norm term in the cost function. Such 1-norm terms are used as regularization term or as penalty for soft constraints (Savorgnan et al., 2011). Furthermore, we also provide the optimal step-size parameter for the algorithm, which is crucial for performance. The convergence rate for the dual function value using the accelerated gradient method is known from Beck and Teboulle (2009), Tseng (2008). This convergence rate in the dual function value does, however, not indicate the rate at which the primal iterate approaches the primal optimal solution. In this chapter we also provide convergence rate results for the primal variables.

Related to this method is the approach presented in Necoara and Suykens (2008) for systems with a (non-strongly) convex cost, which is based on the smoothing technique presented in Nesterov (2005) and makes use of a proximal gradient method. Other relevant work is presented in Kögel and Findeisen (2011), Richter et al. (2009) in which optimization problems arising in MPC are solved in a centralized fashion using accelerated gradient methods. These three methods are, however, restricted to handle only box-constraints on the control signals. The accelerated proximal gradient method presented in this chapter is able to handle arbitrary linear constraints.

The chapter is organized as follows. In Section 3.2, the problem setup is introduced. The dual problem to be solved is formulated in Section 3.3 and some properties of the dual function are presented. The distributed solution algorithm for the dual problem is presented in Section 3.4, followed by the convergence properties of the algorithm in Section 3.5. Section 3.6 concludes this chapter.

3.2 Problem setup

In this chapter we present a distributed algorithm for optimization problems of the form

$$\begin{aligned} \min_{x} \quad & J(x) \triangleq \frac{1}{2} x^T H x + g^T x + \gamma \|Px - p\|_1 \\ \text{s.t.} \quad & A_1 x = B_1 \\ & A_2 x \leq B_2 \end{aligned} \tag{3.1}$$

The full decision vector, $x \in \mathbb{R}^n$, is composed of local decision vectors, $x_i \in \mathbb{R}^{n_i}$, according to $x = [x_1^T, \ldots, x_M^T]^T$. The quadratic cost matrix $H \in \mathbb{R}^{n \times n}$ is assumed separable, i.e. $H = \text{blkdiag}(H_1, \ldots, H_M)$ where $H_i \in \mathbb{R}^{n_i \times n_i}$, the operator blkdiag stands for constructing a block-diagonal matrix from the elemental matrices. This quadratic term in the cost function encompasses the standard MPC cost as in (1.7). The linear part $g \in \mathbb{R}^n$ consists of local parts, $g = [g_1^T, \ldots, g_M^T]^T$ where $g_i \in \mathbb{R}^{n_i}$. Further $P \in \mathbb{R}^{m \times n}$ is composed of $P = [P_1, \ldots, P_m]^T$ where each $P_r \in \mathbb{R}^n$ which in turn consists of $P_r = [P_{r1}^T, \ldots, P_{rM}^T]^T$ with each $P_{ri} \in \mathbb{R}^{n_i}$. The matrix P is not necessarily block-diagonal, which means that the cost function J is not separable. However, we do assume that the vectors P_r have sparse structure. Sparsity refers to the property that for each $r \in \{1, \ldots, m\}$, we have $P_{ri} = 0$ for most of the indices $i \in \{1, \ldots, M\}$. We also have $\gamma > 0$ and $p = [p_1, \ldots, p_m]^T$ with $p_1, \ldots, p_m$ scalars. This gives the following equivalent formulation of the cost function of (3.1)

$$J(x) = \sum_{i=1}^{M} \left[\frac{1}{2} x_i^T H_i x_i + g_i^T x_i \right] + \sum_{r=1}^{m} \left| \sum_{i=1}^{M} P_{ri}^T x_i - p_r \right|. \tag{3.2}$$

The constraint matrices $A_1 \in \mathbb{R}^{q \times n}$ and $A_2 \in \mathbb{R}^{(s-q) \times n}$ contain $a_l \in \mathbb{R}^n$ as $A_1 = [a_1, \ldots, a_q]^T$ and $A_2 = [a_{q+1}, \ldots, a_s]^T$. Further each $a_l = [a_{l1}^T, \ldots, a_{lM}^T]^T$ where $a_{li} \in \mathbb{R}^{n_i}$. Further we have $B_1 \in \mathbb{R}^q$ and $B_2 \in \mathbb{R}^{s-q}$ where $B_1 = [b_1, \ldots, b_q]^T$ and $B_2 = [b_{q+1}, \ldots, b_s]^T$, with $b_1, \ldots, b_s \in \mathbb{R}$. We assume that the matrices A_1 and A_2 are sparse.

By introducing the auxiliary variables x_a and the constraint $Px - p = x_\text{a}$ we get the following optimization problem:

$$\begin{aligned} \min_{x, x_\text{a}} \quad & \frac{1}{2} x^T H x + g^T x + \gamma \|x_\text{a}\|_1 \\ \text{s.t.} \quad & A_1 x = B_1 \\ & A_2 x \leq B_2 \\ & Px - p = x_\text{a} \end{aligned} \tag{3.3}$$

Remark 3.1 *Problem* (3.3) *is more general than problem* (2.1)*, due to the presence of a linear term and a 1-norm term in the cost function, and the last equality constraint.*

The objective of the optimization routine is to solve (3.3) in a distributed fashion using several computational units, where each computational unit computes the optimal local variables, x_i^*, only. Each computational unit is assigned a number of constraints in (3.3) that it is responsible for. This is similar to the way we assign the responsibility of each local controller in Section 2.3.1 of Chapter 2. We denote the set of equality constraints that unit i is responsible for by $\mathfrak{L}_i^1$, the set of inequality constraints by $\mathfrak{L}_i^2$, and the set of constraints originating from the 1-norm by $\mathcal{R}_i$. This division is obviously not unique, but all constraints should be assigned to one computational unit. Further for $l \in \mathfrak{L}_i^1$ and $l \in \mathfrak{L}_i^2$ we require that $a_{li} \neq 0$ and for $r \in \mathcal{R}_i$ that $P_{ri} \neq 0$. The neighborhood set for each

computational unit i can be defined using the following relation:

$$\begin{aligned}\mathcal{N}^i = \big\{ j \in \{1,\dots,M\} \mid\ & \exists l \in \mathfrak{L}_i^1 \text{ s.t. } a_{lj} \neq 0 \text{ or } \exists l \in \mathfrak{L}_j^1 \text{ s.t. } a_{li} \neq 0 \\ & \text{or } \exists l \in \mathfrak{L}_i^2 \text{ s.t. } a_{lj} \neq 0 \text{ or } \exists l \in \mathfrak{L}_j^2 \text{ s.t. } a_{li} \neq 0 \\ & \text{or } \exists r \in \mathcal{R}_i \text{ s.t. } P_{rj} \neq 0 \text{ or } \exists r \in \mathcal{R}_j \text{ s.t. } P_{ri} \neq 0 \big\}\end{aligned}$$

Remark 3.2 *Since we focus on the optimization aspect in this chapter, the neighborhood sets are formulated based on the couplings appearing in the centralized optimization problem. Note that the same neighborhood sets can also be obtained by analyzing the couplings in the dynamics and in the constraints of the MPC problem, as described in Section 1.2 in Chapter 1. Hereby we provide the formulation of neighborhood sets from the optimization problem structure.*

Through the introduction of these sets the constraints that are assigned to unit i can equivalently be written as

$$a_l^T x = b_l \Leftrightarrow \sum_{j \in \mathcal{N}^i} a_{lj}^T x_j = b_l, \qquad l \in \mathfrak{L}_i^1 \tag{3.4}$$

$$a_l^T x \leq b_l \Leftrightarrow \sum_{j \in \mathcal{N}^i} a_{lj}^T x_j \leq b_l, \qquad l \in \mathfrak{L}_i^2 \tag{3.5}$$

and the component inside the 1-norm term can equivalently be written as

$$P_r^T x - p_r = \sum_{j \in \mathcal{N}^i} P_{rj}^T x_j - p_r, \qquad r \in \mathcal{R}_i \tag{3.6}$$

In the following section, the dual function to be maximized is introduced. First, we state some assumptions that are necessary for results in this chapter.

Assumption 3.1 *We assume that each H_i in (3.2) is a real symmetric positive definite matrix that satisfies the following eigenvalue bounds*

$$\underline{\sigma}_i I \preceq H_i \preceq \bar{\sigma}_i I$$

where $0 < \underline{\sigma}_i \leq \bar{\sigma}_i < \infty$.

Remark 3.3 *If Assumption 3.1 holds, the corresponding bound for H becomes $\underline{\sigma} I \preceq H \preceq \bar{\sigma} I$ where $\underline{\sigma} := \min_i \underline{\sigma}_i$ and $\bar{\sigma} := \max_i \bar{\sigma}_i$. Also note that since H is positive definite we have $\frac{1}{\underline{\sigma}} I \preceq H^{-1} \preceq \frac{1}{\bar{\sigma}} I$ (Horn and Johnson, 1990, Corollary 7.7.4).*

Assumption 3.2 *We assume that there exists a vector $\bar{x}$ such that $A_1 \bar{x} = b_1$ and $A_2 \bar{x} < b_2$, this means $\bar{x}$ is a Slater vector (Bertsekas, 1999, Proposition 3.3.9) of problem (3.1). Further, we assume that $a_l, l = 1,\dots,q$ and $P_r, r = 1,\dots,m$ are linearly independent.*

Remark 3.4 *Assumption 3.2 is the Slater condition also for (3.3), since we can choose $\bar{x}_\mathrm{a} = P\bar{x} - p$, and the point $(\bar{x}, \bar{x}_\mathrm{a})$ satisfies the equality constraints and strictly satisfies the inequality constraints of (3.3).*

3.3 Dual problem

In this section we introduce a dual problem to (3.3) from which the primal solution can be obtained. We show that this dual problem has the properties required to apply accelerated gradient methods. We also present some additional properties that are needed to prove convergence rates for the primal variables.

3.3.1 Formulation of the dual problem

We introduce Lagrange multipliers[1], $\lambda \in \mathbb{R}^q, \mu \in \mathbb{R}^{s-q}_{\geq 0}$, $v \in \mathbb{R}^m$ for the constraints in (3.3). Under Assumption 3.2 it is well known (Boyd and Vandenberghe, 2004, §5.2.3) that there is no duality gap and we can obtain the following dual problem

$$\sup_{\lambda,\mu\geq 0,\nu} \inf_{x,x_\mathrm{a}} \Big\{ \frac{1}{2} x^T H x + g^T x + \gamma \|x_\mathrm{a}\|_1 + \lambda^T (A_1 x - B_1) + \\ + \mu^T (A_2 x - B_2) + \nu^T (Px - p - x_\mathrm{a}) \Big\}. \tag{3.7}$$

Rearranging the terms and changing $\inf_x(\cdot)$ to $-\sup_x(-(\cdot))$ leads to

$$\sup_{\lambda,\mu\geq 0,\nu} \Big\{ -\sup_x \Big[-(A_1^T\lambda + A_2^T\mu + P^T\nu + g)^T x - \frac{1}{2} x^T H x \Big] \\ - \lambda^T B_1 - \mu^T B_2 - \nu^T p - \sup_{x_\mathrm{a}} \left[\nu^T x_\mathrm{a} - \gamma \|x_\mathrm{a}\|_1 \right] \Big\}. \tag{3.8}$$

The supremum over x_a can be solved explicitly:

$$\begin{aligned} \sup_{x_\mathrm{a}} \left\{ \nu^T x_\mathrm{a} - \gamma \|x_\mathrm{a}\|_1 \right\} &= \sup_{x_\mathrm{a}} \left\{ \sum_i \left[\nu^i x_\mathrm{a}^i - \gamma |x_\mathrm{a}^i| \right] \right\} \\ &= \sum_i \left\{ \sup_{x_\mathrm{a}^i} \left[\nu^i x_\mathrm{a}^i - \gamma |x_\mathrm{a}^i| \right] \right\} \\ &= \begin{cases} 0 & \text{if } \|\nu\|_\infty \leq \gamma \\ \infty & \text{else} \end{cases} \end{aligned}$$

where the superscript i indicates the i-th element in the vector. The supremum over x_a becomes a box-constraint for the dual variables ν. This is crucial for distribution of the algorithms to be presented in this chapter, since we have a dual problem with only box-constraints that allows gradient-based proximal gradient methods to be applied, where the gradient iterations can be distributed.

Before we explicitly solve the maximization over x in (3.8) the following notation is intro-

[1] The same dual problem can also be formed using Fenchel's duality. In this formulation we use Lagrange's duality that is easier to follow.

duced

$$\mathcal{A} = [A_1^T \ A_2^T \ P^T]^T \qquad \mathcal{B} = [B_1^T \ B_2^T \ p^T]^T \qquad z = [\lambda^T \ \mu^T \ \nu^T]^T$$

where $\mathcal{A} \in \mathbb{R}^{(s+m)\times n}$, $\mathcal{B} \in \mathbb{R}^{s+m}$, and $z \in \mathbb{R}^{s+m}$. We also introduce the set of feasible dual variables

$$Z = \left\{ z \in \mathbb{R}^{s+m} \;\middle|\; \begin{array}{ll} z_l \in \mathbb{R} & l \in \{1,\dots,q\} \\ z_l \geq 0 & l \in \{q+1,\dots,s\} \\ |z_l| \leq \gamma & l \in \{s+1,\dots,s+m\} \end{array} \right\} \tag{3.9}$$

Completion of squares in the maximization over x in (3.8) gives

$$\sup_x \left[-(\mathcal{A}^T z + g)^T x - \frac{1}{2} x^T H x \right] = \frac{1}{2}(\mathcal{A}^T z + g)^T H^{-1}(\mathcal{A}^T z + g)$$

and we get the following dual problem

$$\sup_{z \in Z} \left\{ -\frac{1}{2}(\mathcal{A}^T z + g)^T H^{-1}(\mathcal{A}^T z + g) - \mathcal{B}^T z \right\}. \tag{3.10}$$

We introduce the following definition of the negative dual function

$$f(z) := \frac{1}{2}(\mathcal{A}^T z + g)^T H^{-1}(\mathcal{A}^T z + g) + \mathcal{B}^T z \tag{3.11}$$

It is easily seen that f is convex and differentiable with the following gradient:

$$\nabla f(z) = \mathcal{A} H^{-1}(\mathcal{A}^T z + g) + \mathcal{B}. \tag{3.12}$$

Next we show some properties of the dual problem.

3.3.2 Properties of the dual problem

The dual function has the following properties:

Proposition 3.1 *The gradient, ∇f, is Lipschitz continuous on Z, i.e., there exists a value $L > 0$ such that for any $z_1 \in Z$ and $z_2 \in Z$ we have*

$$\|\nabla f(z_1) - \nabla f(z_2)\|_2 \leq L \|z_1 - z_2\|_2. \tag{3.13}$$

Further, the Lipschitz constant $L = \|\mathcal{A} H^{-1} \mathcal{A}^T\|_2$ is the smallest constant such that (3.13) holds for all $z_1 \in Z$ and $z_2 \in Z$.

Proof. For convenience we introduce $H_{\mathcal{A}} = \mathcal{A} H^{-1} \mathcal{A}^T$. We have

$$\|\nabla f(z_1) - \nabla f(z_2)\|_2 = \|H_{\mathcal{A}}(z_1 - z_2)\|_2 \leq \|H_{\mathcal{A}}\|_2 \|z_1 - z_2\|_2$$

due to the Cauchy-Schwarz inequality. This shows that (3.13) holds. Next we show that $L = \|H_{\mathcal{A}}\|_2$ is the smallest Lipschitz constant on Z. From the definition (3.9) of Z we

conclude that there exist $z_1 \in Z$ and $z_2 \in Z$ such that the difference $d_z = z_1 - z_2$ is parallel to the eigen-vector $v_{\max}(H_{\mathcal{A}})$ corresponding to the largest eigenvalue $\lambda_{\max}(H_{\mathcal{A}})$. Let $\|d_z\|_2 = \epsilon$, by choosing $v_{\max}(H_{\mathcal{A}})$ such that $\|v_{\max}(H_{\mathcal{A}})\| = 1$ we get for some $z_1, z_2 \in Z$:

$$\begin{aligned}\|\nabla f(z_1) - \nabla f(z_2)\|_2 &= \|H_{\mathcal{A}}(z_1 - z_2)\|_2 = \|H_{\mathcal{A}} d_z\|_2 \\ &= \|H_{\mathcal{A}} v_{\max}(H_{\mathcal{A}})\epsilon\|_2 \\ &= \lambda_{\max}(H_{\mathcal{A}})\epsilon \|v_{\max}(H_{\mathcal{A}})\|_2 \\ &= \|H_{\mathcal{A}}\|_2 \epsilon = \|H_{\mathcal{A}}\|_2 \|d_z\|_2 \\ &= \|H_{\mathcal{A}}\|_2 \|z_1 - z_2\|_2\end{aligned}$$

Since the equality can be attained, there is no Lipschitz constant that is smaller than $\|H_{\mathcal{A}}\|_2$. This completes the proof. □

Remark 3.5 *Note that it is known from (Nesterov, 2005, Theorem 1) that ∇f is Lipschitz continuous. However, the Lipschitz constant provided in Nesterov (2005), $\bar{L} = \frac{1}{\sigma}\|\mathcal{A}\|^2$, is larger than the one presented here. We will see in the next section that a smaller Lipschitz constant will lead to a faster convergence rate of the algorithm.*

We also present the following results on the set of optimal dual variables:

Proposition 3.2 *Let Z^* denote the set of optimal dual variables, defined as*

$$Z^* = \{z^* \in Z \mid f(z^*) \leq f(z)\ \forall z \in Z\}$$

The following statements hold:

1. *Z^* is non-empty and bounded*
2. *For any $z^* \in Z^*$ and any $z \in Z$, we have*

$$f(z) - f(z^*) \geq \frac{1}{2\bar{\sigma}}\|\mathcal{A}^T(z - z^*)\|_2^2. \tag{3.14}$$

Proof. Under Assumption 3.2 it is well known (Rockafellar, 1970, Corollary 28.2.2) that the set Z^* is non-empty. Furthermore, from Gauvin (1977) we know that Assumption 3.2 is equivalent to the Mangasarian-Fromovitz Constraint Qualification (MFCQ). In Gauvin (1977) it is shown that MFCQ is equivalent to Z^* being bounded. This proves statement 1.

To prove statement 2 we introduce $f_1(y) = \frac{1}{2}y^T H^{-1} y$, which gives $f(z) = f_1(\mathcal{A}^T z + g) + \mathcal{B}^T z$. From Remark 3.3 we know that $H^{-1} \succeq \frac{1}{\bar{\sigma}} I$ and hence that f_1 is strongly convex and satisfies (cf. (Nesterov, 2004, Definition 2.1.2))

$$f_1(y_1) \geq f_1(y_2) + \langle \nabla f_1(y_2), y_1 - y_2 \rangle + \frac{1}{2\bar{\sigma}}\|y_1 - y_2\|^2$$

We set $y_1 = \mathcal{A}^T z + g$ for any $z \in Z$ and $y_2 = \mathcal{A}^T z^* + g$ for any $z^* \in Z^*$. This gives

$$\begin{aligned}
f(z) &= f_1(\mathcal{A}^T z + g) + \mathcal{B}^T z \\
&\geq f_1(\mathcal{A}^T z^* + g) + \langle \nabla f_1(\mathcal{A}^T z^* + g), \mathcal{A}^T z + g - \mathcal{A}^T z^* - g \rangle \\
&\quad + \frac{1}{2\bar{\sigma}} \|\mathcal{A}^T z + g - \mathcal{A}^T z^* - g\|_2^2 + \mathcal{B}^T z + \mathcal{B}^T (z^* - z^*) \\
&= f(z^*) + \langle \mathcal{A} \nabla f_1(\mathcal{A}^T z^* + g) + \mathcal{B}, z - z^* \rangle + \frac{1}{2\bar{\sigma}} \|\mathcal{A}^T (z - z^*)\|_2^2 \\
&= f(z^*) + \langle \nabla f(z^*), z - z^* \rangle + \frac{1}{2\bar{\sigma}} \|\mathcal{A}^T (z - z^*)\|_2^2 \\
&\geq f(z^*) + \frac{1}{2\bar{\sigma}} \|\mathcal{A}^T (z - z^*)\|_2^2
\end{aligned}$$

where the last inequality comes from the first-order optimality condition for convex functions (cf. (Nesterov, 2004, Theorem 2.2.5))

$$\langle \nabla f(z^*), z - z^* \rangle \geq 0.$$

for any $z \in Z$ and $z^* \in Z^*$. This completes the proof. □

Let us summarize the use of these propositions in the following section. Proposition 3.1 provides the smallest Lipschitz constant of ∇f that will allow choosing the optimal step size of the algorithm, and it helps to guarantee the convergence of the dual variable to the dual solution. Proposition 3.2 will be used when proving the convergence rates for the primal variables in the proximal gradient algorithms.

Remark 3.6 *The Lipschitz continuity property of ∇f means that for any $z_1, z_2 \in \mathbb{R}^{s+m}$ the following holds (cf. Nesterov (2004))*

$$f(z_1) \leq f(z_2) + \langle \nabla f(z_2), z_1 - z_2 \rangle + \frac{L}{2} \|z_1 - z_2\|_2^2 \tag{3.15}$$

An interpretation of (3.15) is that L is the (smallest) curvature of a quadratic function that is tangent to f at z_2 and is an upper bound to the function f for all z_1.

3.4 Distributed proximal gradient algorithms for the dual problem

In this section we present two proximal gradient methods and their distributed versions that minimize the convex dual function f in (3.11) with different convergence rate properties. The first method is a classical proximal gradient method that has convergence rate $O(\frac{1}{k})$, i.e., the distance between the cost function generated in each step and the optimal cost function decreases in the order of $\frac{1}{k}$. The second method is an accelerated proximal gradient method that improves the theoretical convergence rate compared to the standard proximal gradient method with one order of magnitude, i.e., to $O(\frac{1}{k^2})$.

3.4.1 Distributed classical dual proximal gradient method

The proximal gradient algorithm is described by the following iteration:

$$z^{k+1} = \arg\min_{u\in\mathbb{R}^{s+m}} \Big(\delta_Z(u) + f\big(z^k\big) + \langle \nabla f\big(z^k\big), u - z^k\rangle + \frac{1}{2t}\|u - z^k\|_2^2\Big)$$

where t is the step size, k is the iteration index and $\delta_Z(\cdot)$ is the indicator function for the set Z, i.e., $\delta_Z(u) = 0$ if $u \in Z$ and $\delta_Z(u) = \infty$ otherwise. This iteration can equivalently be written as

$$z^{k+1} = \arg\min_{u\in Z} \Big(f\big(z^k\big) + \langle \nabla f\big(z^k\big), u - z^k\rangle + \frac{1}{2t}\|u - z^k\|_2^2\Big).$$

It was shown e.g., in Nesterov (2004) that convergence is guaranteed with the step size parameter $t \in (0, 1/L)$ where L is the smallest Lipschitz constant of ∇f. It was also shown that the step size that gives the fastest theoretical convergence rate is $t = 1/L$. Recall that Proposition 3.1 provides the smallest L, thus $t = 1/L$ is the optimal choice of the step size, and therefore we will use this step size for the remainder of the chapter. We then get

$$z^{k+1} = \arg\min_{u\in Z} \Big(f\big(z^k\big) + \langle \nabla f\big(z^k\big), u - z^k\rangle + \frac{L}{2}\|u - z^k\|_2^2\Big) \tag{3.16}$$

where the function to be minimized is exactly the right-hand side of (3.15). Thus, from the discussion in Remark 3.6 we conclude that the best z^{k+1} is the one that minimizes, over Z, the best quadratic upper bound to f that is tangent to f at z^k.

By removing $f(z^k)$ from (3.16) since it does not affect the minimizing argument, the iteration (3.16) can equivalently be rewritten as

$$\begin{aligned} z^{k+1} &= \arg\min_{u\in Z} \Big(\frac{L}{2}\Big\|u - z^k + \frac{1}{L}\nabla f\big(z^k\big)\Big\|_2^2 - \frac{1}{2L}\|\nabla f(z^k)\|_2^2\Big) \\ &= \arg\min_{u\in Z} \Big(\frac{L}{2}\Big\|u - z^k + \frac{1}{L}\nabla f\big(z^k\big)\Big\|_2^2\Big) \\ &= \mathcal{P}_Z\Big(z^k - \frac{1}{L}\nabla f\big(z^k\big)\Big) \end{aligned} \tag{3.17}$$

where $\mathcal{P}_Z$ is the Euclidean projection onto the set Z. Thus, the new iterate, z^k, is the previous iterate plus a step in the negative gradient direction projected on the feasible set.

Denoting the primal iteration $x^k = H^{-1}(-\mathcal{A}^T z^k - g)$ and by inserting the formula of ∇f from Proposition 3.1, we get the following equivalent iteration

$$x^k = H^{-1}(-\mathcal{A}^T z^k - g) \tag{3.18}$$

$$z^{k+1} = \mathcal{P}_Z\Big(z^k + \frac{1}{L}(\mathcal{A}x^k - \mathcal{B})\Big) \tag{3.19}$$

We recall that the constraint set Z, specified in (3.9), contains only upper and lower bounds on some variables and that the dual variables are partitioned according to $z = [\lambda^T \ \mu^T \ \nu^T]^T$. Then we see that the iteration (3.18)-(3.19) can be parallelized:

$$x^k = H^{-1}(-\mathcal{A}^T z^k - g) \tag{3.20}$$

$$\lambda_l^{k+1} = \lambda_l^k + \frac{1}{L}(a_l^T x^k - b_l) \tag{3.21}$$

$$\mu_l^{k+1} = \max\{0, \mu_l^k + \frac{1}{L}(a_l^T x^k - b_l)\} \tag{3.22}$$

$$\nu_r^{k+1} = \min\{\gamma, \max[-\gamma, \nu_r^k + \frac{1}{L}(P_r^T x^k - p_r)]\}. \tag{3.23}$$

Remark 3.7 *Note that if the 1-norm term $\gamma\|Px - p\|_1$ does not exist in the original problem (3.1), the classical proximal gradient algorithm is still applicable with each iteration consisting of (3.20)–(3.22), and we see that these are the same computations as in Han's parallel algorithm for positive definite quadratic programs (see Algorithm 2.2). This observation shows the similarity between Han's algorithm and the classical proximal gradient algorithm, and they are equivalent for positive definite quadratic programs. However, by studying the classical proximal gradient algorithm, we can obtain its convergence rate, which is not available with Han's method.*

This algorithm still requires global communication since x^k is used in the updates of all dual variables. However, due to the sparse structure of a_l and P_r we will see that only part of the primal update x^k is needed in the updates of dual variables. Before we clarify this statement, we first show how the primal update x^k can be computed in a distributed fashion such that node i computes x_i^k only. Let us now partition the constraint matrix as

$$\mathcal{A} = [\mathcal{A}_1, \ldots, \mathcal{A}_M]$$

where each $\mathcal{A}_i = [a_{1i}, \ldots, a_{si}, P_{1i}, \ldots, P_{mi}]^T \in \mathbb{R}^{(s+m)\times n_i}$ with $a_{li} \in \mathbb{R}^{n_i}, l = 1, \ldots, s$ and $P_{ri} \in \mathbb{R}^{n_i}, r = 1, \ldots, m$. Recall that H is block-diagonal with block diagonal entries $H_i \in \mathbb{R}^{n_i \times n_i}$, the local primal variables are updated according to

$$\begin{aligned} x_i^k &= H_i^{-1}(-\mathcal{A}_i^T z^k - g_i) \\ &= -H_i^{-1}\Big(\sum_{j\in\mathcal{N}^i}\Big[\sum_{l\in\mathfrak{L}_j^1} a_{li}\lambda_l^k + \sum_{l\in\mathfrak{L}_j^2} a_{li}\mu_l^k + \sum_{r\in\mathcal{R}_j} P_{ri}\nu_r^k\Big] + g_i\Big) \end{aligned} \tag{3.24}$$

Thus, each local primal update, x_i^k, can be computed after communication with neighbors $j \in \mathcal{N}^i$.

Before we state the distributed algorithm we recall that

$$a_l^T x^k = \sum_{j\in\mathcal{N}^i} a_{lj}^T x_j^k, \qquad l \in \mathfrak{L}_i^1, l \in \mathfrak{L}_i^2 \tag{3.25}$$

$$P_r^T x^k = \sum_{j\in\mathcal{N}^i} P_{rj}^T x_j^k, \qquad r \in \mathcal{R}_i. \tag{3.26}$$

The resulting algorithm is described below.

Algorithm 3.3 ***Distributed classical proximal gradient algorithm*** *(DCPG)*

Initialize λ^0, μ^0 *and* ν^0.
In every node, i*, the following computations are performed in parallel and with synchronization:*
For $k \geq 0$

1. *Compute*

$$x_i^k = -H_i^{-1}\Big(\sum_{j\in\mathcal{N}^i}\Big[\sum_{l\in\mathfrak{L}_j^1} a_{li}\lambda_l^k + \sum_{l\in\mathfrak{L}_j^2} a_{li}\mu_l^k + \sum_{r\in\mathcal{R}_j} P_{ri}\nu_r^k\Big] + g_i\Big) \tag{3.27}$$

2. *Send* x_i^k *to each* $j \in \mathcal{N}^i$*, receive* x_j^k *from each* $j \in \mathcal{N}^i$

3. *Compute*

$$\lambda_l^{k+1} = \lambda_l^k + \frac{1}{L}\Big(\sum_{j\in\mathcal{N}^i} a_{lj}^T x_j^k - b_l\Big),\ l \in \mathfrak{L}_i^1 \tag{3.28}$$

$$\mu_l^{k+1} = \max\Big\{0, \mu_l^k + \frac{1}{L}\Big(\sum_{j\in\mathcal{N}^i} a_{lj}^T x_j^k - b_l\Big)\Big\},\ l \in \mathfrak{L}_i^2 \tag{3.29}$$

$$\nu_r^{k+1} = \min\Big\{\gamma, \max\Big[-\gamma, \nu_r^k + \frac{1}{L}\Big(\sum_{j\in\mathcal{N}^i} P_{rj}^T x_j^k - p_r\Big)\Big]\Big\},\ r \in \mathcal{R}_i \tag{3.30}$$

4. *Send* $\{\lambda_l^{k+1}\}_{l\in\mathfrak{L}_i^1}, \{\mu_l^{k+1}\}_{l\in\mathfrak{L}_i^2}$ *and* $\{\nu_r^{k+1}\}_{r\in\mathcal{R}_i}$ *to each* $j \in \mathcal{N}^i$*, receive* $\{\lambda_l^{k+1}\}_{l\in\mathfrak{L}_j^1}, \{\mu_l^{k+1}\}_{l\in\mathfrak{L}_j^2}$ *and* $\{\nu_r^{k+1}\}_{r\in\mathcal{R}_j}$ *from each* $j \in \mathcal{N}^i$.

5. *Each local controller checks the local termination criteria*[2]*. If local termination criteria are satisfied, the algorithm stops, otherwise increase* k *and go to step 1) to start a new iteration.*

The convergence result of Algorithm 3.3 will be formulated in Section 3.5.

3.4.2 Distributed accelerated proximal gradient algorithm

In this section we show how the accelerated gradient method can be used to distributively solve (3.3) by minimizing the negative dual function f. The accelerated proximal gradient method for problem (3.10) is defined by the following iteration as presented in (Tseng,

[2] In Chapter 2, we discuss about checking local termination criteria that can also be applied here.

2008, Algorithm 2) and (Beck and Teboulle, 2009, Eq. (4.1)–(4.3))

$$v^k = z^k + \frac{k-1}{k+2}(z^k - z^{k-1}) \tag{3.31}$$

$$z^{k+1} = \mathcal{P}_Z\left(v^k - \frac{1}{L}\nabla f(v^k)\right) \tag{3.32}$$

where $\mathcal{P}_Z$ is the Euclidean projection onto the set Z. Thus, the new iterate, z^{k+1}, is the previous iterate plus a step in the negative gradient direction projected onto the feasible set.

We define the primal iteration $x^k := H^{-1}(-\mathcal{A}^T z^k - g)$. Using this definition, straightforward insertion of v^k into (3.12) gives

$$\nabla f(v^k) = -\mathcal{A}\left(x^k + \frac{k-1}{k+2}(x^k - x^{k-1})\right) + \mathcal{B}$$

By defining $\bar{x}^k = x^k + \frac{k-1}{k+2}(x^k - x^{k-1})$ and recalling the partition $z = [\lambda^T\ \mu^T\ \nu^T]^T$ and the definition (3.9) of the set Z, we find that (3.31)-(3.32) can be parallelized:

$$x^k = H^{-1}(-\mathcal{A}^T z^k - g) \tag{3.33}$$

$$\bar{x}^k = x^k + \frac{k-1}{k+2}(x^k - x^{k-1}) \tag{3.34}$$

$$\lambda_l^{k+1} = \lambda_l^k + \frac{k-1}{k+2}(\lambda_l^k - \lambda_l^{k-1}) + \frac{1}{L}(a_l^T \bar{x}^k - b_l) \tag{3.35}$$

$$\mu_l^{k+1} = \max\left\{0, \mu_l^k + \frac{k-1}{k+2}(\mu_l^k - \mu_l^{k-1}) + \frac{1}{L}(a_l^T \bar{x}^k - b_l)\right\} \tag{3.36}$$

$$\nu_r^{k+1} = \min\left\{\gamma, \max\left[-\gamma, \nu_r^k + \frac{k-1}{k+2}(\nu_r^k - \nu_r^{k-1}) + \frac{1}{L}(P_r^T \bar{x}^k - p_r)\right]\right\}. \tag{3.37}$$

Similarly to the classical proximal gradient version, the step (3.33) can be computed in parallel by (3.24), the computations (3.35)–(3.37) are distributed with the definitions of the sets $\mathfrak{L}_i^1, \mathfrak{L}_i^2, \mathcal{R}_i, i = 1, \ldots, M$ that have been mentioned in Section 3.1. Also note that (3.34) can be computed component-wise. Thus, each local primal update, x_i^k, can be computed after communication with neighbors $j \in \mathcal{N}^i$. Through (3.4)-(3.6) we note that the dual variable iterations can also be updated after communication with neighbors $i \in \mathcal{N}^i$. We get the following distributed algorithm.

Algorithm 3.4 ***Distributed accelerated proximal gradient (DAPG) algorithm***

Initialize $\lambda^0 = \lambda^{-1}, \mu^0 = \mu^{-1}, \nu^0 = \nu^{-1}$ *and* $x^0 = x^{-1}$
In every node, i, the following computations are performed in parallel and with synchronization:
For $k \geq 0$

1. *Compute* x_i^k *according to* (3.24) *and set*

$$\bar{x}_i^k = x_i^k + \frac{k-1}{k+2}(x_i^k - x_i^{k-1})$$

2. *Send $\bar{x}_i^k$ to each $j \in \mathcal{N}^i$, receive $\bar{x}_j^k$ from each $j \in \mathcal{N}^i$*

3. *Compute λ_l^{k+1} according to (3.35) for each $l \in \mathfrak{L}_i^1$, with $a_l^T \bar{x}^k$ computed by (3.4)*
   Compute μ_l^{k+1} according to (3.36) for each $l \in \mathfrak{L}_i^2$, with $a_l^T \bar{x}^k$ computed by (3.5)
   Compute ν_l^{k+1} according to (3.37) for each $l \in \mathcal{R}_i$, with $P_r^T \bar{x}^k - p_r$ computed by (3.6)

4. *Send $\{\lambda_l^{k+1}\}_{l\in\mathfrak{L}_i^1}, \{\mu_l^{k+1}\}_{l\in\mathfrak{L}_i^2}, \{\nu_r^{k+1}\}_{r\in\mathcal{R}_i}$ to each $j \in \mathcal{N}^i$,*
   receive $\{\lambda_l^{k+1}\}_{l\in\mathfrak{L}_j^1}, \{\mu_l^{k+1}\}_{l\in\mathfrak{L}_j^2}$ and $\{\nu_r^{k+1}\}_{r\in\mathcal{R}_j}$ from each $j \in \mathcal{N}^i$

5. *Each local controller checks the local termination criteria. If local termination criteria are satisfied, the algorithm stops, otherwise increase k and go to step 1) to start a new iteration.*

In the next section, we will discuss the convergence results of the DAPG algorithm.

3.5 Properties of the distributed proximal gradient algorithms

In the following theorem we characterize the convergence rates for the dual function and the primal variables using Algorithms 3.3 and 3.4.

Theorem 3.5 *Let Assumptions 3.1 and 3.2 hold. Let L be a Lipschitz constant of ∇f, and $\bar{\sigma}$ and $\underline{\sigma}$ are maximum and minimum eigenvalues of H, respectively. Algorithm 3.3 has the following convergence rate properties:*

1. *Denote z^* as an optimizer of the dual problem (3.10). The generated cost function converges to the optimum according to the bound:*

$$f(z^k) - f(z^*) \leq \frac{L\|z^0 - z^*\|_2^2}{2k}, \forall k \geq 1 \tag{3.38}$$

2. *Denote x^* as the unique optimizer of the primal problem. The generated primal variable converges to the optimizer according to the bound:*

$$\|x^k - x^*\|_2^2 \leq \frac{\bar{\sigma} L\|z^0 - z^*\|_2^2}{\underline{\sigma} k}, \forall k \geq 1 \tag{3.39}$$

Proof.

For the centralized proximal gradient method with iteration (3.16), in which L is the smallest Lipschitz constant of $\nabla f(z)$ as pointed out in Proposition 3.1, the proof for argument 1 has been given in (Toh and Yun, 2010, Theorem 2.1) as well as (Beck and Teboulle, 2009, Theorem 3.1). We will therefore prove that (3.27)–(3.30) generate the same dual updates as (3.16).

As indicated by (3.24), by stacking all x_k^i, $i = 1, \ldots, M$, generated by (3.27) we get x_k as computed by (3.20). Moreover, due to (3.25)–(3.26) we see that (3.28)–(3.30) give the same $\lambda_l^{k+1}, \mu_l^{k+1}, \nu_l^{k+1}$ as (3.21)–(3.23). Thus the computations (3.27)–(3.30) are equivalent with (3.20)–(3.23), which were shown earlier to be the parallel version of (3.16). Consequently, Algorithm 3.3 is a distributed implementation of (3.16), giving the same $z^k, \forall k$. This proves (3.38).

For statement 2 we first show that $x^* = H^{-1}(-\mathcal{A}^T z^* - g)$. KKT conditions for the quadratic programs (Boyd and Vandenberghe, 2004, p. 244) imply that the primal optimal solution, x^*, and dual optimal solutions z^* must satisfy

$$0 = Hx^* + g + \mathcal{A}^T z^* \Leftrightarrow x^* = H^{-1}(-\mathcal{A}^T z^* - g)$$

since H is invertible. This leads to

$$\begin{aligned}\|x^k - x^*\|_2^2 &= \|H^{-1}(\mathcal{A}^T z^k - \mathcal{A}^T z^*)\|_2^2 \\ &\leq \|H^{-1}\|_2^2 \|\mathcal{A}^T z^k - \mathcal{A}^T z^*\|_2^2 \\ &\leq \frac{1}{\underline{\sigma}^2}\|\mathcal{A}^T z^k - \mathcal{A}^T z^*\|_2^2 \\ &\leq \frac{2\bar{\sigma}}{\underline{\sigma}^2}(f(z^k) - f(z^*)) \leq \frac{\bar{\sigma} L \|z^0 - z^*\|_2^2}{\underline{\sigma} k}\end{aligned}$$

where the second inequality is based on Remark 3.3, the third inequality on Proposition 3.2, and the final inequality is from (3.38).

This completes the proof. □

The rate of convergence, $O(\frac{1}{k})$, achieved with Algorithm 3.3 is sub-optimal compared to what can be achieved by gradient methods. With Algorithm 3.4, we obtain a better convergence rate, namely $O(\frac{1}{k^2})$, as stated in the following theorem.

Theorem 3.6 *Let Assumptions 3.1 and 3.2 hold. Let L be a Lipschitz constant of ∇f, and $\bar{\sigma}$ and $\underline{\sigma}$ are maximum and minimum eigenvalues of H, respectively. Algorithm 3.4 has the following convergence rate properties:*

1. *Denote an optimizer of the dual problem (3.10) as z^*. The generated cost function converges to the optimum according to the bound:*

$$f(z^k) - f(z^*) \leq \frac{2L\|z^0 - z^*\|_2^2}{(k+1)^2}, \forall k \geq 1 \tag{3.40}$$

2. *Denote the unique optimizer of the primal problem as x^*. The generated primal variable converges to the optimizer according to the bound:*

$$\|x^k - x^*\|_2^2 \leq \frac{4\bar{\sigma} L\|z^0 - z^*\|_2^2}{\underline{\sigma}^2(k+1)^2}, \forall k \geq 1 \tag{3.41}$$

Proof. The derivation of Algorithm 3.4 shows that it is a distributed implementation of (Tseng, 2008, Algorithm 2) and (Beck and Teboulle, 2009, Eq. 4.1-4.3) applied to mini-

mize f. The bound in statement 1 follows from (Tseng, 2008, Proposition 2) and (Beck and Teboulle, 2009, Theorem 4.4).

The proof for statement 2 is similar to the proof of (3.39), with the use of (3.41) instead of (3.38) in the last inequality. The inequality chain is as follows:

$$\begin{aligned}\|x^k - x^*\|_2^2 &= \|H^{-1}(\mathcal{A}^T z^k - \mathcal{A}^T z^*)\|_2^2 \\ &\leq \|H^{-1}\|^2 \|\mathcal{A}^T z^k - \mathcal{A}^T z^*\|_2^2 \\ &\leq \frac{1}{\underline{\sigma}^2}\|\mathcal{A}^T z^k - \mathcal{A}^T z^*\|_2^2 \\ &\leq \frac{2\bar{\sigma}}{\underline{\sigma}^2}(f(z^k) - f(z^*)) \leq \frac{4\bar{\sigma}L\|z^0 - z^*\|_2^2}{\underline{\sigma}^2(k+1)^2}.\end{aligned}$$

where the second inequality is based on Remark 3.3, the third inequality on Proposition 3.2 and the final inequality is from (3.40). This completes the proof. □

3.6 Conclusions

We have presented a distributed optimization algorithm for strongly convex optimization problems with sparse problem data. The algorithm is based on an accelerated gradient method that is applied to the dual problem. We have shown that the distributed algorithm provides a fast convergence rate on the cost function and a corresponding convergence rate is drawn for the primal variable. The proposed DAPG method can be applied to distributed MPC with similar subsystem decomposition as when we apply distributed Han's method, moreover the DAPG can also deal with the MPC problem where there are mixed 1-norm and 2-norm terms in the cost function, making it suitable for a larger problem class, e.g., reference-tracking MPC problems where the objective of reference-tracking is characterized by a 1-norm term in the cost function. The hydro power valley application problem that we will treat in detail in Chapter 5 belongs to this class of problems.

Chapter 4

Distributed model predictive control with guaranteed feasibility

This chapter deals with an issue related to the asymptotic convergence of dual decomposition approaches, which often do not provide a feasible solution of the primal optimization problem in a finite number of iterations. Our idea is to use a constraint tightening approach, and then apply a primal-dual iterative algorithm that provides bounds on the constraint violation and the suboptimality. We develop two primal-dual iterative algorithms, leading to two hierarchical MPC methods that provide feasible solutions in every sampling step, one approach is based on a hierarchical conjugate gradient method, the other is based on a distributed Jacobi algorithm. Closed-loop stability is established using bounded suboptimality.

4.1 Introduction

In Chapters 2 and 3, we have presented two distributed schemes based on dual decomposition for solving large-scale MPC problems with coupling in both dynamics and constraints. A typical requirement of the dual decomposition-based methods is that the dual problem needs to be solved exactly. However, the distributed algorithms we proposed in Chapters 2 and 3 are developed upon iterative approaches that only converge asymptotically to the optimum, which may lead to difficulties when attempting to implement these approaches in a real-time environment. In the current chapter we present a novel method that is motivated by the use of constraint tightening in robust MPC (Kuwata et al., 2007). This method allows terminating the iterations for the dual problem before reaching convergence while still guaranteeing a feasible primal solution to be found. The proposed approach in this chapter uses a projected gradient method for maximizing the dual function, along with a primal averaging scheme that yields bounds on constraint violation and the cost function. With this approach, we develop two algorithms: one is based on a hierarchical conjugate gradient method, and the other is based on a distributed Jacobi method for solving the primal problem within each iteration of the projected gradient method. The two resulting algorithms have a two-layer iteration structure where most of the computation tasks are

done by local controllers, while a few crucial parameters are handled by a coordinator. The proposed framework guarantees primal feasible solutions and MPC stability using a finite number of iterations with bounded suboptimality.

This chapter is organized as follows. In Section 4.2, we recall the MPC optimization problem in this thesis and present its tightened version, which will be used to guarantee feasibility of the original problem even with a suboptimal primal solution. Sections 4.3 and 4.4 provide the main elements of the two algorithms used to solve the dual version of the tightened optimization problem, together with the properties of the algorithms: we show that the primal average solution generated by the approximate subgradient algorithm is a feasible solution of the original optimization problem, and that the cost function decreases through the MPC updates. This allows it to be used as a Lyapunov function for showing closed-loop MPC stability. Additional discussion about the realization of the assumptions is provided in Section 4.5. A simulation example is provided in Section 4.6 to illustrate the performance of the proposed framework. Section 4.7 concludes the chapter and outlines future research.

4.2 Problem description

4.2.1 MPC problem formulation

Recall that we have M interconnected subsystems with coupled discrete-time linear time-invariant dynamics:

$$x_{k+1}^i = \sum_{j=1}^{M} A^{ij} x_k^j + B^{ij} u_k^j, \quad i = 1, \dots, M \tag{4.1}$$

and the corresponding centralized state-space model:

$$x_{k+1} = Ax_k + Bu_k \tag{4.2}$$

with $x_k = [(x_k^1)^T (x_k^2)^T \dots (x_k^M)^T]^T$, $u_k = [(u_k^1)^T (u_k^2)^T \dots (u_k^M)^T]^T$, $A = [A_{ij}]_{i,j\in\{1,\dots,M\}}$ and $B = [B_{ij}]_{i,j\in\{1,\dots,M\}}$.

We formulate an MPC problem using a terminal penalty and a terminal constraint set. In a particular time step t the MPC optimization problem is defined as follows:

$$\min_{\mathbf{u},\mathbf{x}} \quad \sum_{k=t}^{t+N-1} \left(x_k^T Q x_k + u_k^T R u_k \right) + x_{t+N}^T P x_{t+N} \tag{4.3}$$

$$\text{s.t.} \quad x_{k+1}^i = \sum_{j\in\mathcal{N}^i} A^{ij} x_k^j + B^{ij} u_k^j,$$

$$i = 1, \dots, M, \quad k = t, \dots, t+N-1 \tag{4.4}$$

$$x_k \in \mathcal{X}, k = t+1, \dots, t+N-1 \tag{4.5}$$

$$x_{t+N} \in X_\mathrm{f} \subset \mathcal{X} \tag{4.6}$$

$$u_k \in \mathcal{U}, k = t, \ldots, t+N-1 \tag{4.7}$$
$$u_k^i \in \Omega_i, i = 1, \ldots, M, \quad k = t, \ldots, t+N-1 \tag{4.8}$$
$$x_t = x(t) \in \mathcal{X} \tag{4.9}$$

where $\mathbf{u} = [u_t^T, \ldots, u_{t+N-1}^T]^T$, $\mathbf{x} = [x_{t+1}^T, \ldots, x_{t+N}^T]^T$, the matrices Q, P, and R are block-diagonal and positive definite, the constraint sets $\mathcal{U}$, $\mathcal{X}$, and X_f are polytopes and have nonempty interiors, and each local constraint set Ω_i is a hyperbox. Each subsystem i is assigned a neighborhood, denoted $\mathcal{N}^i$, containing subsystems that have direct dynamical interactions with subsystem i, including itself. The initial state x_t is the current state at time step t. Notice that the MPC formulation (4.3)–(4.9) does not incorporate the terminal zero-point constraint ($x_{t+N} = 0$) like the MPC problem introduced in (1.10)–(1.14). Moreover, the approach proposed in this chapter does not handle any equality constraints on the states except the dynamical constraints (4.4), since those corresponding constraint sets will have empty interior that prevent constraint tightening, the key idea of this approach.

As $\mathcal{U}$, $\mathcal{X}$, and X_f are polytopes, the constraints (4.5) and (4.6) are represented by linear inequalities. Moreover, the state vector $\mathbf{x}$ is affinely dependent on $\mathbf{u}$. Hence, we can eliminate the state variables $x_{t+1}, \ldots, x_{t+N}$ and transform the constraints (4.4), (4.5), and (4.6) into linear inequalities of the input variable $\mathbf{u}$. Eliminating the state variables in (4.3)–(4.9) leads to an optimization problem in the following form:

$$f_t^* = \min_{\mathbf{u}} \quad f(\mathbf{u}, x_t) \tag{4.10}$$
$$\text{s.t.} \quad g(\mathbf{u}, x_t) \leq 0 \tag{4.11}$$
$$\mathbf{u} \in \boldsymbol{\Omega} \tag{4.12}$$

where f and $g = [g_1, \ldots, g_m]^T$ are convex functions, and $\boldsymbol{\Omega} = \Omega \times \cdots \times \Omega$ (N times) with $\Omega = \prod_{i=1}^M \Omega_i$, is a hyperbox. Note that $f(\mathbf{u}, x_t) > 0, \forall \mathbf{u} \neq 0, x_t \neq 0$, due to the positive definiteness of Q, P, and R.

Following the standard approach of the *dual mode MPC* formulation (Mayne et al., 2000), we assume that the following assumptions hold:

Assumption 4.1 *There exists a block-diagonal feedback gain K such that the matrix $A + BK$ is Schur*[1] *(i.e., K yields a decentralized stabilizing control law for the unconstrained aggregate system).*

Assumption 4.2 *The terminal constraint set X_f is strictly positively invariant for the closed-loop system with $x_{k+1} = (A + BK)x_k$, i.e., if $x \in \mathrm{int}(X_\mathrm{f})$ then $(A + BK)x \in \mathrm{int}(X_\mathrm{f})$. In addition, for any state in X_f, the control input generated by the terminal controller should satisfy the input constraints, i.e., $-Kx \in \mathcal{U} \cap \Omega$, $\forall x \in X_\mathrm{f}$.*

Remark 4.1 *Assumptions 4.1 and 4.2 are necessary to ensure the system will be stable when its state is driven into the positively invariant set X_f and the MPC controller is switched to the linear controller $u = -Kx$. Similar assumptions are typically used in the dual mode MPC approach, however here we need two stronger conditions. The first one*

[1] A matrix is Schur if all of its eigenvalues are inside the unit circle.

is the block-diagonal structure of the matrix K that means the existence of decentralized linear stabilizing controllers within X_{f}. The second one is that once the linear controller $u = -Kx$ will keep the state in the interior of X_{f} for any starting state inside the interior of X_{f}, thus also requires that X_{f} has nonempty interior, this requirement will be used for updating the Slater vector, which is introduced in the following assumption.

Assumption 4.3 *The Slater condition holds for problem (4.11)–(4.12), i.e., there exists a vector that satisfies (4.11)–(4.12) with strict inequality constraints (Bertsekas, 1999). It is also assumed that prior to each time step t, a Slater vector $\bar{\mathbf{u}}_t$ is available, such that*

$$\begin{cases} g_j(\bar{\mathbf{u}}_t, x_t) < 0, j = 1, \ldots, m \\ \bar{\mathbf{u}}_t \in \mathrm{int}(\mathbf{\Omega}) \end{cases} \tag{4.13}$$

Remark 4.2 *The Slater condition means that the union of the constraint sets after eliminating state variables in (4.3)–(4.9) has nonempty interior. If this condition holds, we will only need to find the Slater vector $\bar{\mathbf{u}}_0$ for the first time step, which can be computed offline. In Section 4.5.1 we will show that a new Slater vector can then be obtained for each $t \geq 1$, using Assumption 4.2.*

Assumption 4.4 *At each time step t, the following holds*

$$f(\mathbf{u}_{t-1}, x_{t-1}) > f(\bar{\mathbf{u}}_t, x_t) \tag{4.14}$$

For later reference, we define $\Delta_t > 0$ which can be computed before time step t as follows:

$$\Delta_t = f(\mathbf{u}_{t-1}, x_{t-1}) - f(\bar{\mathbf{u}}_t, x_t) \tag{4.15}$$

Remark 4.3 *Assumption 4.4 is often satisfied with an appropriate terminal penalty matrix P. A method to construct a block-diagonal P with a given decentralized stabilizing control law is provided in Šiljak (1978). Often the terminal state that is inside the set X_{f} and the subsequent input generated by the terminal controller $u = -Kx$ both have small magnitudes, hence if $\bar{\mathbf{u}}_t$ is constructed by shifting $\mathbf{u}_{t-1}$ one step ahead, we can obtain Δ_t approximately as:*

$$\Delta_t \simeq x_{t-1}^T Q x_{t-1} + u_{t-1}^T R u_{t-1} \tag{4.16}$$

Assumption 4.5 *For each $x_t \in \mathcal{X}$, the Euclidean norm of $g(\mathbf{u}, x_t)$ is bounded:*

$$\exists L_t : L_t \geq \|g(\mathbf{u}, x_t)\|_2, \forall \mathbf{u} \in \mathbf{\Omega} \tag{4.17}$$

Remark 4.4 *In the first time step, with given x_0, we can find L_0 by evaluating $\|g(\mathbf{u}, x_0)\|_2$ at the vertices of $\mathbf{\Omega}$; the maximum will then satisfy (4.17) for $t = 0$, due to the convexity of g and $\mathbf{\Omega}$. For the subsequent time steps, we will present a simple method to update L_t in Section 4.5.2.*

The main focus in this chapter is to solve problem (4.10)–(4.12) in a distributed or hierarchical manner by using dual decomposition approaches, while guaranteeing feasibility

and stability of the closed-loop system. Let us denote $(\mathbf{u}_t, \mathbf{x}_t)$ as a feasible solution generated by the controller for problem (4.3)–(4.9) at time step t. This solution is required to be feasible but not necessarily optimal. In the remaining of this chapter, we will present two approaches to obtain feasible $(\mathbf{u}_t, \mathbf{x}_t)$ at every sampling time, while also guaranteeing stability of the MPC closed-loop system. Both approaches aim to tighten the constraints that are (4.11)–(4.12) for the first method and only (4.11) for the second method, then solve the dual problem using a gradient method. The differences between the two approaches come from the use of different techniques to solve the minimization of the Lagrangian, which is required when solving the dual problem.

We now present the tightened problem, which serves as the starting point for both approaches.

4.2.2 The tightened problem

In our proposed approach to the presented challenges, we will not solve problem (4.10)–(4.12) directly. Instead, we will first tighten the problem (4.10)–(4.12) before applying iterative algorithms. Next we describe two ways for constraint tightening that will be used in two different algorithms.

Tightening of both coupled and decoupled constraints:

Let us combine the two constraints (4.11)–(4.12) into one common form:

$$h(\mathbf{u}, x_t) \leq 0 \tag{4.18}$$

with $h = [h_1, \ldots, h_{m+2n_\mathbf{u}}]^T$ where $h_j = g_j, j = 1, \ldots, m$, and $h_{m+1}, \ldots, h_{m+2n_\mathbf{u}}$ represent the domain constraint (4.12), note that h_j is a linear function for every j, due to the linearity of g and $\mathbf{\Omega}$ is a convex polytopic set.

With a given Slater vector $\bar{\mathbf{u}}_t$ as mentioned in Assumption 4.3, we can pick a value c_t such that:

$$0 < c_t < \min_{j=1,\ldots,m+2n_\mathbf{u}} \{-h_j(\bar{\mathbf{u}}_t, x_t)\} \tag{4.19}$$

and form the tightened problem:

$$f_t'^* = \min_{\mathbf{u}} \quad f(\mathbf{u}, x_t) \tag{4.20}$$

$$\text{s.t.} \quad h'(\mathbf{u}, x_t) \leq 0 \tag{4.21}$$

with the new constraint function h' defined as:

$$h'(\mathbf{u}, x_t) \triangleq h(\mathbf{u}, x_t) + \mathbf{1}_{m+2n_\mathbf{u}} c_t \tag{4.22}$$

where $\mathbf{1}_{m+2n_\mathbf{u}}$ is a column vector with $m + 2n_\mathbf{u}$ entries that are all equal to 1.

Due to (4.13) and (4.19), we have

$$\max_{j=1,\ldots,m+2n_\mathbf{u}} \{h'_j(\bar{\mathbf{u}}_t, x_t)\} = \max_{j=1,\ldots,m+2n_\mathbf{u}} \{h_j(\bar{\mathbf{u}}_t, x_t)\} + c_t < 0 \tag{4.23}$$

Hence $h'_j(\bar{\mathbf{u}}_t, x_t) < 0, j = 1, \ldots, m + 2n_{\mathbf{u}}$, this means $\bar{\mathbf{u}}_t$ is also a Slater vector for the constraint (4.21).

Tightening of coupled constraints only:

In the second case, we consider the following tightened problem:

$$\tilde{f'}_t^* = \min_{\mathbf{u}} \quad f(\mathbf{u}, x_t) \tag{4.24}$$

$$\text{s.t.} \quad g'(\mathbf{u}, x_t) \leq 0 \tag{4.25}$$

$$\mathbf{u} \in \mathbf{\Omega} \tag{4.26}$$

with the tightened constraint:

$$g'(\mathbf{u}, x_t) \triangleq g(\mathbf{u}, x_t) + \mathbf{1}_m \tilde{c}_t \tag{4.27}$$

where $g'(\mathbf{u}, x_t) = [g'_1, \ldots, g'_m]^T$ and

$$0 < \tilde{c}_t < \min_{j=1,\ldots,m} \{-g_j(\bar{\mathbf{u}}_t, x_t)\} \tag{4.28}$$

Due to (4.13) and (4.28), we have

$$\max_{j=1,\ldots,m} \{g'_j(\bar{\mathbf{u}}_t, x_t)\} = \max_{j=1,\ldots,m} \{g_j(\bar{\mathbf{u}}_t, x_t)\} + \tilde{c}_t < 0 \tag{4.29}$$

and this also confirms that $\bar{\mathbf{u}}_t$ is a Slater vector for (4.25). Moreover, using (4.17) and the triangle inequality of the 2-norm, we will get $\tilde{L'}_t = L_t + \tilde{c}_t$ as the norm bound for g', i.e., $\tilde{L'}_t \geq \|g'(\mathbf{u}, x_t)\|_2, \forall \mathbf{u} \in \mathbf{\Omega}$. Note that $\tilde{L'}_t$ implicitly depends on x_t, as $\bar{\mathbf{u}}_t$ and $\tilde{c}_t$ are updated based on the current state x_t.

Remark 4.5 *Due to the fact that h includes g and additional decoupled constraints, comparing (4.19) and (4.28) we see that $\max c_t \leq \max \tilde{c}_t$. This means the range for tuning c_t is narrower than the range for tuning $\tilde{c}_t$, i.e., the tightening approach for the function h is more conservative than tightening g only. It will be shown in the next section that by tightening also the local constraints, the resulting dual problem can be treated with more algorithms.*

Now we have two tightened optimization problems that are (4.20)–(4.21) and (4.24)–(4.26). Note that due to the dynamical and constraint couplings in the MPC formulation, both these problems are large-scale quadratic optimization problems, and we can write the functions as

$$f(\mathbf{u}, x_t) = \mathbf{u}^T H \mathbf{u} + b^T(x_t)\mathbf{u} \tag{4.30}$$

$$h'(\mathbf{u}, x_t) = C\mathbf{u} - d(x_t) + 1_{m+2n_{\mathbf{u}}} c_t \tag{4.31}$$

$$g'(\mathbf{u}, x_t) = \tilde{C}\mathbf{u} - \tilde{d}(x_t) + 1_m \tilde{c}_t \tag{4.32}$$

where H is a positive definite matrix, b, d, and $\tilde{d}$ are constant vectors depending on the initial state value, and both H and C have sparse structure resulting from the interconnection of subsystems.

4.2.3 Dual problem formulations

We consider solving the dual problem of (4.20)–(4.21) or (4.24)–(4.26), since the dual decomposition approach allows dealing with the coupled constraints in a hierarchical way.

The exact dual problem

The Lagrangian of problem (4.20)–(4.21) is defined as

$$\mathcal{L}(\mathbf{u}, \mu) = f(\mathbf{u}) + \mu^T h'(\mathbf{u}) \tag{4.33}$$

in which $\mu \in \mathbb{R}_+^{m+2n_\mathbf{u}}$ is called the dual variable.

The dual function of (4.20)–(4.21),

$$q'(\mu) = \min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \mu) \tag{4.34}$$

is a concave function. Note that if f and h' are continuous functions, and if $\mathcal{L}(\cdot, \mu)$ attains the minimum at a unique point $\mathbf{u}(\mu)$, then according to (Bertsekas, 1999, Proposition 6.1.1), q' is differentiable everywhere and

$$\nabla q'(\mu) = h'(\mathbf{u}(\mu)), \quad \forall \mu \in \mathbb{R}_+^{m+2n_\mathbf{u}} \tag{4.35}$$

Given the assumption that the Slater condition holds for (4.20)–(4.21), duality theory (Bertsekas, 1999, Chapter 5) shows that

$$q_t'^* = f_t'^* \tag{4.36}$$

where $q_t'^* = \max_{\mu \in \mathbb{R}_+^{m+2n_\mathbf{u}}} q'(\mu)$ and $f_t'^*$ is the minimum of (4.20)–(4.21).

The inexact dual problem

Now we consider the dual problem of (4.24)–(4.26), which has the following Lagrangian:

$$\tilde{\mathcal{L}}(\mathbf{u}, \mu) = f(\mathbf{u}) + \mu^T g'(\mathbf{u}) \tag{4.37}$$

and the dual function is defined as:

$$\tilde{q'}(\mu) = \min_{\mathbf{u} \in \mathbf{\Omega}} \tilde{\mathcal{L}}(\mathbf{u}, \mu) \tag{4.38}$$

In case an optimum of the Lagrangian is not attained due to termination of the optimization algorithm after a finite number of steps, a value $\tilde{\mathbf{u}}(\bar{\mu})$ that satisfies

$$\tilde{\mathcal{L}}(\tilde{\mathbf{u}}(\bar{\mu}), \bar{\mu}) \leq \min_{\mathbf{u} \in \mathbf{\Omega}} \tilde{\mathcal{L}}(\mathbf{u}, \bar{\mu}) + \delta \tag{4.39}$$

will lead to the following inequality:

$$\tilde{q'}(\mu) \leq \tilde{q'}(\bar{\mu}) + \delta + (\mu - \bar{\mu})^T g'(\tilde{\mathbf{u}}(\bar{\mu})), \quad \forall \mu \in \mathbb{R}_+^m \tag{4.40}$$

where $g'(\tilde{\mathbf{u}}(\bar{\mu}))$ is called a δ-subgradient of the dual function $\tilde{q'}$ at the point $\bar{\mu}$. The set of all δ-subgradients of $\tilde{q'}$ at $\bar{\mu}$ is called δ-subdifferential of $\tilde{q'}$ at $\bar{\mu}$.

This also means we do not have to look for a subgradient (or δ-subgradient) of the dual function, as it is available by just evaluating the constraint function at the primal value $\mathbf{u}(\bar{\mu})$ (or $\tilde{\mathbf{u}}(\bar{\mu})$).

In the next sections, we will present a projected gradient method for solving the exact dual problem of (4.20)–(4.21), and then another method for solving the inexact dual problem of (4.24)–(4.26).

4.3 Hierarchical MPC using a conjugate gradient method (HPF-DEG)

In this section, we focus on the solution of the optimization problem (4.20)–(4.21), which needs to be calculated in each MPC update step. For simplicity of exposition, the dependence of functions on x_t will be omitted. The main idea is to solve the exact dual problem of (4.20)–(4.21).

4.3.1 Projected gradient method

The projected gradient iteration for solving (4.20)–(4.21) is given by

$$\mathbf{u}^{(k)} = \arg\min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \mu^{(k)}) \tag{4.41}$$

$$\mu^{(k+1)} = \mathcal{P}_{\mathbb{R}_+^{m+2n_\mathbf{u}}} \left\{ \mu^{(k)} + \alpha_t h'^{(k)} \right\} \tag{4.42}$$

where k stands for the iteration index, the operator $\mathcal{P}_{\mathbb{R}_+^{m+2n_\mathbf{u}}}$ is the projection onto the nonnegative orthant, α_t is the constant step size (used for time step t), $\mu^{(k)}$ is the dual iterate at iteration k (for the first iteration, $\mu^{(0)} = 0 \cdot \mathbf{1}_m$), and $h'^{(k)} = h'\left(\mathbf{u}^{(k)}, x_t\right)$ is the gradient of the dual function $q'(\mu)$ at iteration k.

The step size α_t for the dual update (4.42) should satisfy the inequality:

$$\alpha_t \leq \frac{2\Delta_t}{L'^2_t} \tag{4.43}$$

where Δ_t is provided by (4.15), and L'_t is the norm bound for $h'^{(k)}$. This step size is chosen to facilitate showing the monotonic decrease of $f(\mathbf{u}_t, x_t)$ in Section 4.3.4.

The projected gradient iteration (4.41)–(4.42) is performed for $k = 1, \ldots, \bar{k}_t$, with $\bar{k}_t \in \mathbb{Z}_+$ defined a priori as

$$\bar{k}_t \geq \frac{1}{\alpha_t \epsilon_t} \left(\frac{3}{\lambda_t} f(\bar{\mathbf{u}}_t) + \frac{\alpha_t L'^2_t}{2\lambda_t} + \alpha_t L'_t \right) \tag{4.44}$$

We will show in Section 4.3.4 that once the number of iterations (4.41)–(4.42) exceeds $\bar{k}_t$ steps, then we will obtain a feasible solution for the original problem (4.10)–(4.11) by averaging the primal iterates:

$$\hat{\mathbf{u}}^{(\bar{k}_t)} = \frac{1}{\bar{k}_t} \sum_{l=0}^{\bar{k}_t} \mathbf{u}^{(l)} \tag{4.45}$$

Remark 4.6 *In order to implement the algorithm in a hierarchical fashion, we need a hierarchical method to solve problem* (4.41) *and perform* (4.42)*. Even though* (4.41) *is indeed an unconstrained quadratic optimization problem that has an analytical solution, we will not use this solution due to the computational burden when inverting the Hessian matrix. Instead, we will employ a conjugate gradient method (Bertsekas and Tsitsiklis, 1989, Chapter 2) and use a hierarchical implementation to find the solution of* (4.41)*. The dual update* (4.42) *will also be done locally by letting each constraint be updated by a local controller.*

In the next subsection, we describe a decomposition of the large-scale system, and present the hierarchical conjugate gradient method using the decomposition structure.

4.3.2 Subsystem decomposition

Note that the functions f and h' have particular structure since the matrices Q, P, and R in the cost function (4.10) are block-diagonal. It is straightforward to verify (Venkat et al., 2007) that if matrices A and B are sparse (meaning that the large-scale system consists of subsystems with neighboring interactions only), then H and C will also be sparse, and b and d will have a structured dependence on x_t. Now, consider the following subsystem decomposition:

- Each subsystem $i = 1, \ldots, M$ has an associated decision variable $\mathbf{u}^i$ with the same dimension as $\mathbf{u}$, but containing only the variables corresponding to subsystem i in its nonzero entries[2]. We define $\mathbf{u}^i$ as

$$\mathbf{u}^i = \mathfrak{I}^i \mathbf{u} \tag{4.46}$$

  where $\mathfrak{I}^i \in \mathbb{R}^{n_\mathbf{u} \times n_\mathbf{u}}$ is a diagonal matrix with zeros and ones on its diagonal. Matrices $\mathfrak{I}^i$ (and thus the subsystem decomposition) are chosen such that there is no overlap between the subsystems' variables.

- For each subsystem i, there is a neighborhood $\mathcal{N}^i$ that contains i itself and any other subsystem j that is coupled with i either via the objective function f (i.e., there is at least one term involving both variables of i and $j \in \mathcal{N}^i$ in f), or via the constraint function h' (i.e., there is at least one constraint that involves some variables of i and $j \in \mathcal{N}^i$).

[2] Typically $\mathbf{u}^i$ contains the control inputs of subsystem i over the prediction horizon of the MPC problem.

In order to distribute the dual update (4.42), we will let each subsystem be in charge of updating a subset of dual variables, denoted by $\mathcal{D}^i$. There are different methods for partitioning the dual variables, among them one simple partitioning algorithm is the following: if the maximum absolute value of entries in a row r of C corresponds to a variable of subsystem i, then $r \in \mathcal{D}^i$. Note that each dual variable is updated by one and only one subsystem.

Since $h'(\mathbf{u}) = C\mathbf{u} - d$, we see that in order to perform the update (4.42) for the dual variables within $\mathcal{D}^i$, subsystem i will only need to communicate with subsystems $j \in \mathcal{N}^i$ during iteration k to get the necessary entries of $\mathbf{u}^{(k)}$.

4.3.3 Hierarchical conjugate gradient method

The algorithm we propose to use for solving (4.41) is an adaptation of the conjugate gradient method as described in (Bertsekas and Tsitsiklis, 1989, Chapter 2). Hereby we summarize the main steps and underlying ideas of this iterative method:

- The algorithm starts at some $\mathbf{u}(0)$ and selects $s(0) = -\nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u}(0), \mu^{(k)}) = -(H\mathbf{u}(0) + b(x_t) + C^T\mu^{(k)})$.
- The iteration has the form:

$$\mathbf{u}(p+1) = \mathbf{u}(p) + \gamma(p)s(p),\ p = 0, 1, \ldots \tag{4.47}$$

  with p the (inner) iteration index, $s(p)$ the direction of update at iteration p, and $\gamma(p)$ an optimal scalar step size.
- The algorithm stops if $\nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u}(p), \mu^{(k)}) = 0$. Otherwise, update $s(p)$ by

$$s(p) = -\nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u}(p), \mu^{(k)}) + \beta(p)s(p-1) \tag{4.48}$$

  where $\beta(p)$ is generated by

$$\beta(p) = \frac{\nabla_{\mathbf{u}}^T\mathcal{L}(\mathbf{u}(p), \mu^{(k)})\nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u}(p), \mu^{(k)})}{\nabla_{\mathbf{u}}^T\mathcal{L}(\mathbf{u}(p-1), \mu^{(k)})\nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u}(p-1), \mu^{(k)})} \tag{4.49}$$

- We update $\gamma(p)$ by

$$\gamma(p) = -\frac{s(p)^T\nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u}(p), \mu^{(k)})}{s(p)^T H s(p)} \tag{4.50}$$

- One feature of this iteration method is the conjugate property of $s(p)$, i.e.,

$$s(p)^T H s(r) = 0, \forall r \neq p \tag{4.51}$$

It has been proved that the conjugate gradient algorithm terminates after at most $n_{\mathbf{u}}$ steps[3], where $n_{\mathbf{u}}$ is the size of $\mathbf{u}$ (Bertsekas and Tsitsiklis, 1989, Chapter 2).

[3] The number of iterations can be significantly reduced with proper preconditioning.

For application in hierarchical MPC, it is required that the communications and computations of (4.47), (4.48), (4.49), (4.50), (4.42) and (4.45) can be done in a hierarchical setting. We propose to use a hierarchical optimization method, in which a coordinator communicates with all local controllers, and each local controller can also communicate with others in its neighborhood. In summary, we propose a nested iterative algorithm in which the outer loop is the projected gradient method for the dual problem (4.41)–(4.42), and the inner loop is the hierarchical conjugate gradient method for solving (4.41).

Algorithm 4.1 ***Hierarchical Primal Feasible method with Dual Exact Gradient (HPF-DEG)***

1. *Input:* $\bar{\mathbf{u}}_t, \delta_t, L_t$. *The coordinator computes* c_t, λ_t, α_t, *and* $\bar{k}_t$, *then sends* c_t, α_t *and* $\bar{k}_t$ *to all local controllers.*
2. *Set* $k = 0$. *Choose* $\mu^{(0)} = 0 \cdot \mathbf{1}_{m+2n_{\mathbf{u}}}$, $\mathbf{u}^{(0)} = 0 \cdot \mathbf{1}_{n_{\mathbf{u}}}$, *then each local controller has* ${\mathbf{u}^{(0)}}^i = 0 \cdot \mathbf{1}_{n_{\mathbf{u}}}$.
3. *Solve* (4.41) *at step* k *by the following iterative process:*
   (a) *Set* $p = 0$. *Initialize each local controller* $i \in \{1, \ldots, M\}$ *with* $\mathbf{u}(0)^i = {\mathbf{u}^{(k)}}^i$.
   (b) *Each local controller* $i \in \{1, \ldots, M\}$ *communicates with* $j \in \mathcal{N}^i$ *to get* $\mathbf{u}(p)^j$, *then computes*

$$\nabla \mathcal{L}(\mathbf{u}(p))^i = \mathfrak{I}^i \left(H \sum_{j \in \mathcal{N}^i} \mathbf{u}^j(p) + b + C^T \mu^{(k)} \right)$$

   (c) *Each local controller* $i \in \{1, \ldots, M\}$ *computes* $\nabla \mathcal{L}^T(\mathbf{u}(p))^i \nabla \mathcal{L}(\mathbf{u}(p))^i$, *and then sends the result to the coordinator.*
   (d) *The coordinator makes the sum:*

$$\nabla \mathcal{L}^T(\mathbf{u}(p)) \nabla \mathcal{L}(\mathbf{u}(p)) = \sum_{i=1}^{M} \nabla \mathcal{L}^T(\mathbf{u}(p))^i \nabla \mathcal{L}(\mathbf{u}(p))^i$$

   Note that steps 3(b), 3(c), and 3(d) are aimed at updating $\nabla \mathcal{L}^T(\mathbf{u}(p)) \nabla \mathcal{L}(\mathbf{u}(p))$, *which appears in* (4.49), *by computing its subsystem components.*
   (e) *The coordinator checks whether* $\nabla \mathcal{L}(\mathbf{u}(p)) = 0$. *If so, then it announces "stop" and each controller takes* ${\mathbf{u}^{(k)}}^i = \mathbf{u}(p)^i$. *Go to step (4). If* $\nabla \mathcal{L}(\mathbf{u}(p)) \neq 0$, *the coordinator computes* $\beta(p)$ *by* (4.49), *and sends* $\beta(p)$ *to all local controllers.*
   (f) *Each local controller* $i \in \{1, \ldots, M\}$ *computes* $s(p)^i = -\nabla \mathcal{L}(\mathbf{u}(p))^i + \beta(p) s(p-1)^i$ *if* $p > 0$, *or* $s(p)^i = -\nabla \mathcal{L}(\mathbf{u}(p))^i$ *if* $p = 0$, *and then communicates with* $j \in \mathcal{N}^i$ *to get* $s(p)^j$. *The purpose of this step is to use local implementation for computing* (4.48).
   (g) *Each local controller* $i \in \{1, \ldots, M\}$ *computes* $[s(p)^i]^T \nabla \mathcal{L}(\mathbf{u}(p))^i$ *and also* $[s(p)^i]^T H \sum_{j \in \mathcal{N}^i} s(p)^j$, *and then sends these results to the coordinator.*

(h) The coordinator makes the sums:

$$s(p)^T \nabla \mathcal{L}(\mathbf{u}(p)) = \sum_{i=1}^{M} [s(p)^i]^T \nabla \mathcal{L}(\mathbf{u}(p))^i$$

$$s(p)^T s(p) = \sum_{i=1}^{M} \left\{ [s(p)^i]^T H \sum_{j \in \mathcal{N}^i} s(p)^j \right\},$$

computes $\gamma(p)$ according to (4.50), and then sends $\gamma(p)$ to all local controllers.

(i) Each local controller $i \in \{1, \ldots, M\}$ updates $\mathbf{u}(p+1)^i = \mathbf{u}(p)^i + \gamma(p) s(p)^i$.

(j) Set $p = p + 1$, go to step 3(b).

4. *Each local controller $i \in \{1, \ldots, M\}$ communicates with $j \in \mathcal{N}^i$ to get ${\mathbf{u}^{(k)}}^j$.*

5. *Each local controller $i \in \{1, \ldots, M\}$ updates the dual variables in the set $\mathcal{D}^i$ by:*

$$\mu^{(k+1)}(l) = \mathcal{P}_{\mathbb{R}_+} \left\{ \mu^{(k)}(l) + \alpha_t \left(\sum_{v=1}^{n_{\mathbf{u}}} C(l, v) \mathbf{u}^{(k)}(v) - d(l) \right) \right\}, \forall l \in \mathcal{D}^i$$

where all $C(l, v)$ that are nonzero correspond to variables v of the subsystems $j \in \mathcal{N}^i$, and therefore the knowledge of ${\mathbf{u}^{(k)}}^j, j \in \mathcal{N}^i$ is enough for this computation.

6. *Set $k = k + 1$. If $k \leq \bar{k}_t$, go to step (3).*

7. *Each local controller $i \in \{1, \ldots, M\}$ computes $\hat{\mathbf{u}}^i = \frac{1}{k_t} \sum_{k=0}^{\bar{k}_t} {\mathbf{u}^{(k)}}^i$. The corresponding global output is $\hat{\mathbf{u}}^{(\bar{k}_t)} = \sum_{i=1}^{M} \hat{\mathbf{u}}^i$.*

Remark 4.7 *Algorithm 4.1 needs a coordinator to compute and deliver common variables c_t, α_t and $\bar{k}_t$ to the local controllers. However, most of the computations are carried out by local controllers. Each local controller needs to exchange information with its neighbors and the coordinator. Regarding communications, major communications between subsystems are in the order of $2\bar{k}_t \times n_{\mathbf{u}} \times \sum_{i=1}^{M} |\mathcal{N}^i|$ messages (occuring at steps 3(b) and 4 of Algorithm 4.1), while the communications between the coordinator and all local controllers are in the order of $2\bar{k}_t \times n_{\mathbf{u}} \times M$ messages (occuring at steps 3(c) and 3(g) of Algorithm 4.1).*

In the next section, we will show that $\hat{\mathbf{u}}^{(\bar{k}_t)}$ generated by Algorithm 4.1 is a feasible solution of (4.10)–(4.11), and it ensures cost reduction for the MPC problem.

4.3.4 Properties of the HPF-DEG algorithm

Denoting the primal average sequence by $\hat{\mathbf{u}}^{(k)} = \frac{1}{k} \sum_{l=0}^{k} \mathbf{u}^{(l)}$ where $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(k)}$ are generated by (4.41), we will make use of the following results (Nedic and Ozdaglar, 2009)

for $k \geq 1$:

$$\left\| \left[h'\left(\hat{\mathbf{u}}^{(k)}\right) \right]^+ \right\|_2 \leq \frac{1}{k\alpha_t} \left(\frac{3}{\lambda_t} [f(\bar{\mathbf{u}}_t) - q_t'^*] + \frac{\alpha_t L'^2_t}{2\lambda_t} + \alpha_t L'_t \right) \tag{4.52}$$

$$f\left(\hat{\mathbf{u}}^{(k)}\right) \leq f_t'^* + \frac{\left\|\mu^{(0)}\right\|_2^2}{2k\alpha_t} + \frac{\alpha_t L'^2_t}{2} \tag{4.53}$$

where the notation h'^+ indicates the constraint violation, i.e., $h'^+ = \max\{h', 0 \cdot \mathbf{1}_{m+2n_\mathbf{u}}\}$. Using the constraint violation bound (4.52) and the cost upper bound (4.53) for the tightened problem (4.24)–(4.25), we will show that $\hat{\mathbf{u}}^{(\bar{k}_t)}$ is a feasible solution of (4.10)–(4.11), and $f(\mathbf{u}_t, x_t) \leq f(\mathbf{u}_{t-1}, x_{t-1})$.

Feasible primal solution

Proposition 4.2 *Let Assumptions 4.1–4.5 hold and Algorithm 4.1 be executed until $k = \bar{k}_t$ defined by (4.44). The primal average $\hat{\mathbf{u}}^{(\bar{k}_t)}$ is a feasible solution of (4.10)–(4.11).*

Proof: Applying the result in (4.52) leads to

$$\left\| \left[h'\left(\hat{\mathbf{u}}^{(\bar{k}_t)}\right) \right]^+ \right\|_2 \leq \frac{1}{\bar{k}_t\alpha_t} \left(\frac{3}{\lambda_t} [f(\bar{\mathbf{u}}_t) - q_t'^*] + \frac{\alpha_t L'^2_t}{2\lambda_t} + \alpha_t L'_t \right)$$

Moreover, $q_t'^* = f_t'^* \geq 0$ because $f(\mathbf{u}) \geq 0, \forall \mathbf{u}$ due to the use of a positive definite quadratic stage cost in the MPC setting. We have

$$\left\| \left[h'\left(\hat{\mathbf{u}}^{(\bar{k}_t)}\right) \right]^+ \right\|_2 \leq \frac{1}{\bar{k}_t\alpha_t} \left(\frac{3}{\lambda_t} f(\bar{\mathbf{u}}_t) + \frac{\alpha_t L'^2_t}{2\lambda_t} + \alpha_t L'_t \right) \tag{4.54}$$

Combining (4.54) and (4.44), and noticing that $\bar{k}_t$ and c_t are both positive leads to

$$\left\| \left[h'\left(\hat{\mathbf{u}}^{(\bar{k}_t)}\right) \right]^+ \right\|_2 \leq c_t \tag{4.55}$$

$$\Rightarrow h'_j\left(\hat{\mathbf{u}}^{(\bar{k}_t)}\right) \leq c_t, \quad j = 1, \ldots, m + 2n_\mathbf{u} \tag{4.56}$$

$$\Rightarrow h_j\left(\hat{\mathbf{u}}^{(\bar{k}_t)}\right) < 0, \quad j = 1, \ldots, m + 2n_\mathbf{u} \tag{4.57}$$

in which the last inequality is due to (4.27). This means that $\hat{\mathbf{u}}^{(\bar{k}_t)}$ is a feasible solution of the problem (4.10)–(4.11). □

Decreasing cost function

Let us recall that the optimization problem formulation is motivated by an MPC problem for which Assumption 4.4 holds. The following proposition shows that the cost function of the MPC problem is a decreasing function.

Proposition 4.3 *Let $\mathbf{u}_{t-1}$ and x_{t-1} be given and satisfy (4.14), and Assumptions 4.1–4.5 hold. Considering $\mathbf{u}_t = \hat{\mathbf{u}}^{(\bar{k}_t)}$ generated by Algorithm 4.1, the following inequality holds:*

$$f(\mathbf{u}_t, x_t) \leq f(\mathbf{u}_{t-1}, x_{t-1}) \tag{4.58}$$

Proof. Using (4.53), (4.43), and noting that $\mu^{(0)} = 0$ leads to

$$f(\mathbf{u}_t, x_t) \triangleq f\left(\hat{\mathbf{u}}^{(\bar{k}_t)}\right) \leq f_t'^* + \frac{\alpha_t L_t'^2}{2} \tag{4.59}$$

Due to (4.43), we then have

$$f(\mathbf{u}_t, x_t) \leq f_t'^* + \Delta_t \tag{4.60}$$

Notice that $\bar{\mathbf{u}}_t$ is also a feasible solution of (4.20)–(4.21) (due to the way we construct the tightened problem: $\bar{\mathbf{u}}_t$ still belongs to the interior of the tightened constraint set), while $f_t'^*$ is the optimal cost value of this problem. As a consequence,

$$f_t'^* \leq f(\bar{\mathbf{u}}_t, x_t) \tag{4.61}$$

Combining (4.60), (4.61), and (4.15) results in the cost decrease property: $f(\mathbf{u}_t, x_t) \leq f(\mathbf{u}_{t-1}, x_{t-1})$ □

In summary, the algorithm HPF-DEG is able to generate a feasible solution for the MPC optimization problem at each time step t, which is used to show that the cost function is decreasing thanks to Assumption 4.4. In order to employ this algorithm for hierarchical MPC, one needs to make sure that a feasible prediction $\bar{\mathbf{u}}_t$ of the input sequence, and a cost reduction Δ_t are available before each time step t. We will discuss the method to update $\bar{\mathbf{u}}_t$ in Section 4.5.1, while Δ_t can be computed by (4.15).

4.4 Hierarchical MPC using a distributed Jacobi method (HPF-DAG)

The objective of this section is to use dual decomposition for the tightened problem (4.24)–(4.26). This section differs from the previous one by using an inexact dual problem. We will show that a similar approach as in the previous section can be used, with a distributed Jacobi algorithm instead of the conjugate gradient method in the inner loop, and extending the results for the projected gradient method in the outer loop.

Recall that the tightened problem to be considered in this section is:

$$\tilde{f}_t'^* = \min_{\mathbf{u}} \quad f(\mathbf{u}, x_t) \tag{4.62}$$

$$\text{s.t.} \quad g'(\mathbf{u}, x_t) \leq 0 \tag{4.63}$$

$$\mathbf{u} \in \mathbf{\Omega} \tag{4.64}$$

The Slater condition also holds for the tightened problem (4.62)–(4.64), with $\bar{\mathbf{u}}_t$ being the Slater vector.

4.4.1 Projected gradient method with approximate gradient

With the same subsystem decomposition as described in Section 4.3.2, we organize our algorithm for solving (4.10)–(4.12) at time step t in a nested iteration of an outer and inner loop. The main procedure is described as follows:

Algorithm 4.4 ***Hierarchical Primal Feasible method with Dual Approximate Gradient (HPF-DAG)***

1. *Given a Slater vector $\bar{\mathbf{u}}_t$ of (4.10)–(4.12), determine $\tilde{c}_t$ and construct the tightened problem (4.62)–(4.64).*

2. *Determine the step size $\tilde{\alpha}_t$, the suboptimality ε_t, and $\tilde{k}_t$ (the sufficient number of outer iterations), see later in Section 4.4.3.*

3. ***Outer loop:*** *Set $\mu^{(0)} = 0 \cdot \mathbf{1}_m$. For $k = 0, \ldots, \tilde{k}$, find $\mathbf{u}^{(k)}, \mu^{(k+1)}$ such that:*

$$\tilde{\mathcal{L}}(\mathbf{u}^{(k)}, \mu^{(k)}) \leq \min_{\mathbf{u} \in \mathbf{\Omega}} \tilde{\mathcal{L}}(\mathbf{u}, \mu^{(k)}) + \varepsilon_t \tag{4.65}$$

$$\mu^{(k+1)} = \mathcal{P}_{\mathbb{R}^m_+} \left\{ \mu^{(k)} + \tilde{\alpha}_t d^{(k)} \right\} \tag{4.66}$$

   where $\mathcal{P}_{\mathbb{R}^m_+}$ denotes the projection onto the nonnegative orthant, $d^{(k)} = g'\left(\mathbf{u}^{(k)}, x_t\right)$ is a ε_t-subgradient of the dual function $\tilde{q}'$ at $\mu^{(k)}$ (see (4.40)).

   Inner loop:

   - *Determine $\tilde{p}_k$ (the sufficient number of inner iterations), see later in Section 4.4.5.*
   - *Solve problem (4.65) in a distributed way with a Jacobi algorithm. For $p = 0, \ldots, \tilde{p}_k$, every subsystem i computes:*

$$\begin{aligned} \mathbf{u}^i(p+1) = \arg \min_{\mathbf{u}^i \in \mathbf{\Omega}_i} \tilde{\mathcal{L}}(\mathbf{u}^1(p), \ldots, \mathbf{u}^{i-1}(p), \mathbf{u}^i, \\ \mathbf{u}^{i+1}(p), \ldots, \mathbf{u}^M(p), \mu^{(k)}) \end{aligned} \tag{4.67}$$

     where $\mathbf{\Omega}_i$ is the local constraint set for control variables of subsystem i.

   - *Define $\mathbf{u}^{(k)} \triangleq [\mathbf{u}^1(\tilde{p}_k)^T, \ldots, \mathbf{u}^M(\tilde{p}_k)^T]^T$, which is guaranteed to satisfy (4.65).*

4. *Compute $\hat{\mathbf{u}}^{(\tilde{k}_t)} = \frac{1}{\tilde{k}_t} \sum_{l=0}^{\tilde{k}_t} \mathbf{u}^{(l)}$, and take $\mathbf{u}_t = \hat{\mathbf{u}}^{(\tilde{k}_t)}$ as the solution of (4.10)–(4.12).*

Remark 4.8 *Algorithm 4.4 is suitable for implementation in a hierarchical fashion where the main computations occur in the Jacobi iterations and are executed in parallel by local controllers, while the updates of dual variables and common parameters are carried out by a higher-level coordinating controller. In the inner loop, each subsystem only needs to communicate with its neighbors, which will be discussed in Section 4.4.6. This algorithm is also amenable to implementation in distributed settings, where there are communication links available to help determine and propagate the common parameters* $\tilde{\alpha}_t, \varepsilon_t, \tilde{k}_t$, *and* $\tilde{p}_k$.

Remark 4.9 *Algorithm 4.4 uses a similar projected gradient update as in (4.41)–(4.42) for maximizing the dual function; however, it uses the distributed Jacobi iteration for minimizing the Lagrangian, as opposed to the conjugate gradient method in Section 4.3.*

In the following sections, we will describe further results in terms of an upper bound on the cost function and the convergence rate of the Jacobi iteration, leading to the determination of the parameters of the algorithm.

4.4.2 Extended upper bound for the primal cost function

The outer loop at iteration k uses an approximate subgradient method. Let us formulate the bounds on constraint violation and on the cost function by the next proposition.

Proposition 4.5 *The primal average sequence* $\hat{\mathbf{u}}^{(k)} = \frac{1}{k}\sum_{l=0}^{k} \mathbf{u}^{(l)}$ *generated at iteration* $k \geq 1$ *of the Algorithm 4.4 has the following properties:*

$$\left\| \left[g'\left(\hat{\mathbf{u}}^{(k)}, x_t\right) \right]^+ \right\|_2 \leq \frac{1}{k\tilde{\alpha}_t} \left(\frac{3}{\gamma_t} [f(\bar{\mathbf{u}}_t, x_t) - \tilde{q'}^*_t] + \frac{\tilde{\alpha}_t {L'_t}^2}{2\gamma_t} + \tilde{\alpha} L'_t \right) \tag{4.68}$$

$$f\left(\hat{\mathbf{u}}^{(k)}, x_t\right) \leq \tilde{f'}^*_t + \frac{\left\|\mu^{(0)}\right\|_2^2}{2k\tilde{\alpha}_t} + \frac{\tilde{\alpha}_t {L'_t}^2}{2} + \varepsilon_t \tag{4.69}$$

where ${g'}^+$ *denotes the constraint violation, i.e.,* ${g'}^+ = \max\{g', 0 \cdot \mathbf{1}_m\}$.

Proof. The bound on the constraint (4.68) is from Nedic and Ozdaglar (2009) and was mentioned before at (4.52), while the bound on the cost (4.69) is an extension of (4.53) for the case where ε_t-subgradient is used instead of exact subgradient. Hereby we focus on the proof for (4.69).

This proof is an extension of the proof of Proposition 3(b) in Nedic and Ozdaglar (2009), the main difference being the incorporation of the suboptimality ε_t in the update of the primal variable (4.65).

Using the convexity of the cost function, we have:

$$\begin{aligned} f(\hat{\mathbf{u}}^{(k)}) &= f\left(\frac{1}{k}\sum_{l=0}^{k-1} \mathbf{u}^{(l)}\right) \leq \frac{1}{k}\sum_{l=0}^{k-1} f(\mathbf{u}^{(l)}) \\ &= \frac{1}{k}\sum_{l=0}^{k-1} \left(f(\mathbf{u}^{(l)}) + (\mu^{(l)})^T g'(\mathbf{u}^{(l)}) \right) - \frac{1}{k}\sum_{l=0}^{k-1} (\mu^{(l)})^T g'(\mathbf{u}^{(l)}) \end{aligned} \tag{4.70}$$

Note that $\tilde{\mathcal{L}}\big(\mathbf{u}^{(l)}, \mu^{(l)}\big) = f(\mathbf{u}^{(l)}) + g'(\mathbf{u}^{(l)})^T \mu^{(l)}$ and

$$\tilde{\mathcal{L}}\big(\mathbf{u}^{(l)}, \mu^{(l)}\big) \leq \min_{\mathbf{u}\in\boldsymbol{\Omega}} \tilde{\mathcal{L}}\big(\mathbf{u}^{(l)}, \mu^{(l)}\big) + \varepsilon_t = \tilde{q'}\big(\mu^{(l)}\big) + \varepsilon_t, \forall l < k \tag{4.71}$$

Combining the two inequalities above, we then have:

$$\begin{aligned} f(\hat{\mathbf{u}}^{(k)}) &\leq \frac{1}{k}\sum_{l=0}^{k-1} \tilde{q'}\big(\mu^{(l)}\big) + \varepsilon_t - \frac{1}{k}\sum_{l=0}^{k-1} (\mu^{(l)})^T g'(\mathbf{u}^{(l)}) \\ &\leq \tilde{q'}_t^{\,*} + \varepsilon_t - \frac{1}{k}\sum_{l=0}^{k-1} (\mu^{(l)})^T d^{(l)} \end{aligned} \tag{4.72}$$

where $d^{(l)} = g'(\mathbf{u}^{(l)})$, and the last inequality is due to $\tilde{q'}_t^{\,*} \geq \tilde{q'}\big(\mu^{(l)}\big), \forall l$.

Using the expression of the squared sum:

$$\begin{aligned} \|\mu^{(l+1)}\|_2^2 &\leq \|\mu^{(l)} + \tilde{\alpha}_t d^{(l)}\|_2^2 \\ &= \|\mu^{(l)}\|_2^2 + 2\tilde{\alpha}_t (\mu^{(l)})^T d^{(l)} + \|\tilde{\alpha}_t d^{(l)}\|_2^2 \end{aligned} \tag{4.73}$$

we have:

$$-(\mu^{(l)})^T d^{(l)} \leq \frac{1}{2\tilde{\alpha}_t}\left(\|\mu^{(l)}\|_2^2 - \|\mu^{(l+1)}\|_2^2 + \tilde{\alpha}_t^2 \|d^{(l)}\|_2^2\right) \tag{4.74}$$

for $l = 0, \ldots, k-1$.

Summing side by side for $l = 0, \ldots, k-1$, we get:

$$-\sum_{l=0}^{k-1} (\mu^{(l)})^T d^{(l)} \leq \frac{1}{2\tilde{\alpha}_t}\left(\|\mu^{(0)}\|_2^2 - \|\mu^{(k)}\|_2^2\right) + \frac{\tilde{\alpha}_t}{2}\sum_{l=0}^{k-1} \|d^{(l)}\|_2^2 \tag{4.75}$$

Linking (4.72) and (4.75), we then have:

$$\begin{aligned} f(\hat{\mathbf{u}}^{(k)}) &\leq \tilde{q'}_t^{\,*} + \varepsilon_t + \frac{1}{2k\tilde{\alpha}_t}\left(\|\mu^{(0)}\|^2 - \|\mu^{(k)}\|^2\right) + \frac{\tilde{\alpha}_t}{2k}\sum_{l=0}^{k-1} \|d^{(l)}\|^2 \\ &\leq \tilde{q'}_t^{\,*} + \frac{\|\mu^{(0)}\|^2}{2k\tilde{\alpha}_t} + \frac{\tilde{\alpha}_t {L'_t}^2}{2} + \varepsilon_t \end{aligned} \tag{4.76}$$

in which we get the last inequality by using L'_t as the norm bound for all $g'(\mathbf{u}^{(l)}), l = 0, \ldots, k-1$.

Finally, due to the Slater condition, there is no primal-dual gap, i.e., $\tilde{q'}_t^{\,*} = f_t^*$ (cf. (4.36)), and hence:

$$f(\hat{\mathbf{u}}^{(k)}) \leq f_t'^{*} + \frac{\|\mu^{(0)}\|^2}{2k\tilde{\alpha}_t} + \frac{\tilde{\alpha}_t {L'_t}^2}{2} + \varepsilon_t$$

□

4.4.3 Determining the step size $\tilde{\alpha}_t$, the suboptimality ε_t, and the outer loop termination step $\tilde{k}_t$

Similarly to Section 4.3, we need to determine $\tilde{\alpha}_t$, ε_t, and $\tilde{k}_t$ such that at the end of the algorithm, we will get a feasible solution for problem (4.10)–(4.12), which is the average of primal iterates generated by (4.65):

$$\mathbf{u}_t \triangleq \hat{\mathbf{u}}^{(\tilde{k}_t)} = \frac{1}{\tilde{k}_t} \sum_{l=0}^{\tilde{k}_t} \mathbf{u}^{(l)} \tag{4.77}$$

and the monotonic decrease of the cost function, i.e., $f(\mathbf{u}_t, x_t) < f(\mathbf{u}_{t-1}, x_{t-1})$.

The step size $\tilde{\alpha}_t$ and the suboptimality ε_t should satisfy:

$$\frac{\tilde{\alpha}_t {L'_t}^2}{2} + \varepsilon_t \leq \Delta_t \tag{4.78}$$

where Δ_t is defined in (4.15), and L'_t is the norm bound for g'. This condition allows us to show the monotonic decrease of the cost function in problem (4.3)–(4.9), which can then be used as a Lyapunov function.

Note that a larger $\tilde{\alpha}_t$ will lead to a smaller number of outer iterations, while a larger ε_t will lead to a smaller number of inner iterations. For the remainder of the chapter we choose their values according to

$$\tilde{\alpha}_t = \frac{\Delta_t}{{L'_t}^2} \tag{4.79}$$

$$\varepsilon_t = \frac{\Delta_t}{2} \tag{4.80}$$

The subgradient iteration (4.65)–(4.66) is performed for $k = 1, \ldots, \tilde{k}_t$, with $\tilde{k}_t \in \mathbb{Z}_+$ defined a priori as

$$\tilde{k}_t \geq \frac{1}{\tilde{\alpha}_t c_t} \left(\frac{3}{\gamma_t} f(\bar{\mathbf{u}}_t, x_t) + \frac{\tilde{\alpha}_t {L'_t}^2}{2\gamma_t} + \tilde{\alpha}_t L'_t \right) \tag{4.81}$$

where $\gamma_t = \min_{j=1,\ldots,m}\{-g'_j(\bar{\mathbf{u}}_t, x_t)\} = \min_{j=1,\ldots,m}\{-g_j(\bar{\mathbf{u}}_t, x_t)\} - \tilde{c}_t$, and $\bar{\mathbf{u}}_t$ is the Slater vector of (4.62)–(4.64). This choice of $\tilde{k}_t$ comes from the results in Proposition 4.5, so that after $\tilde{k}_t$ outer iterations, Algorithm 4.4 generates a feasible solution, as will be pointed out in Section 4.4.6.

4.4.4 Convergence rate of the distributed Jacobi algorithm

The inner iteration (4.67) performs parallel local optimizations based on a standard Jacobi distributed optimization method for a convex function $\tilde{\mathcal{L}}(\mathbf{u}, \mu^{(k)})$ over a Cartesian product set $\boldsymbol{\Omega}$, as described in (Bertsekas and Tsitsiklis, 1989, Section 3.3). In order to find a sufficient stopping condition for this Jacobi iteration, we need to characterize the convergence rate of this algorithm.

Firstly, let us define the block-maximum 2-norm of a vector.

Definition 4.1 *For a vector $x = [x_1^T, \ldots, x_M^T]$ with $x_i \in \mathbb{R}^{n_i}$, let $\mathfrak{p}$ denote the tuple $(n_1, \ldots, n_M)$ and let $\|\cdot\|_2$ stand for the Euclidean norm. The block-maximum 2-norm of x with respect to the partitioning $\mathfrak{p}$ is defined as:*

$$\|x\|_{\mathrm{b-m},2,\mathfrak{p}} = \max_i \|x_i\|_2 \tag{4.82}$$

Next we provide the condition for convergence of the Jacobi iteration. Note that since $f(\mathbf{u}, x_t)$ is a strictly convex quadratic function, and $g'(\mathbf{u}, x_t)$ contains only linear functions, the function $\tilde{\mathcal{L}}(\mathbf{u}, \mu^{(k)})$ is also a strictly convex quadratic function with respect to $\mathbf{u}$. Hence, it can be written as:

$$\tilde{\mathcal{L}}(\mathbf{u}, \mu^{(k)}) = \mathbf{u}^T \mathcal{H} \mathbf{u} + b^T \mathbf{u} + c \tag{4.83}$$

where $\mathcal{H}$ is a symmetric, positive definite matrix, b is a constant vector and c is a constant scalar.

Proposition 4.6 *Suppose the following condition holds:*

$$\lambda_{\min}(\mathcal{H}_{ii}) > \sum_{j \neq i} \bar{\sigma}(\mathcal{H}_{ij}), \forall i \tag{4.84}$$

where $\mathcal{H}_{ij}$ with $i, j \in \{1, \ldots, M\}$ denotes the submatrix of the Hessian $\mathcal{H}$ of $\tilde{\mathcal{L}}$ w.r.t. $\mathbf{u}$, containing entries of $\mathcal{H}$ in rows belonging to subsystem i and columns belonging to subsystem j, $\lambda_{\min}$ denotes the smallest eigenvalue, and $\bar{\sigma}$ denotes the maximum singular value.

Then there exists $\phi \in (0, 1)$ such that the aggregate solution of the Jacobi iteration (4.67) satisfies:

$$\|\mathbf{u}(p) - \mathbf{u}^*\|_2 \leq M \phi^p \max_i \|\mathbf{u}^i(0) - \mathbf{u}^{i*}\|_2, \quad \forall p \geq 1 \tag{4.85}$$

where $\mathbf{u}^* = \arg\min_{\mathbf{u} \in \mathbf{\Omega}} \tilde{\mathcal{L}}(\mathbf{u}, \mu^{(k)})$, and $\mathbf{u}^{i*}$ is the component of subsystem i in $\mathbf{u}^*$.

Proof. According to Proposition 3.10 in (Bertsekas and Tsitsiklis, 1989, Chapter 3), the Jacobi algorithm has a linear convergence w.r.t. the block-maximum 2-norm.

Proposition 3.10 in (Bertsekas and Tsitsiklis, 1989, Chapter 3) states that $\mathbf{u}(p)$ generated by (4.67) will converge to the optimizer of $\tilde{\mathcal{L}}(\mathbf{u}, \mu^{(k)})$ with a linear convergence rate w.r.t. the block-maximum 2-norm (i.e. $\|\mathbf{u}(p) - \mathbf{u}^*\|_{\mathrm{b-m},2,\mathfrak{p}} \leq \phi^p \|\mathbf{u}(0) - \mathbf{u}^*\|_{\mathrm{b-m},2,\mathfrak{p}}$, with $\mathbf{u}^* = \arg\min_{\mathbf{u} \in \mathbf{\Omega}} \tilde{\mathcal{L}}(\mathbf{u}, \mu^{(k)})$ and $\phi \in [0, 1)$) if there exists a positive scalar γ such that the mapping $R : \mathbf{\Omega} \to \mathbb{R}^{n_\mathbf{u}}$, defined by $R(\mathbf{u}) = \mathbf{u} - \gamma \nabla_\mathbf{u} \tilde{\mathcal{L}}(\mathbf{u}, \mu^{(k)})$, is a contraction w.r.t. the block-maximum 2-norm.

Our focus now is to derive a condition such that $R(\mathbf{u})$ is a contraction mapping.

In order to derive a condition for $R(\mathbf{u})$ to be a contraction mapping, we will make use of Proposition 1.10 in (Bertsekas and Tsitsiklis, 1989, Chapter 3) for the 2-norm case, stating that:

If $f : \mathbb{R}^{n_{\mathbf{u}}} \to \mathbb{R}^{n_{\mathbf{u}}}$ is continuously differentiable and there exists a scalar $\phi \in [0, 1)$ such that

$$\|I - \gamma G_i^{-1}\big(\nabla_i F_i(\mathbf{u})\big)^T\|_2 + \sum_{j \neq i} \|\gamma G_i^{-1}\big(\nabla_j F_i(\mathbf{u})\big)^T\|_2 \leq \phi, \forall \mathbf{u} \in \mathbf{\Omega}, \forall i \tag{4.86}$$

then the mapping $T : \mathbf{\Omega} \to \mathbb{R}^{n_{\mathbf{u}}}$ defined by $T_i(\mathbf{u}) = \mathbf{u}_i - \gamma G_i^{-1} F(\mathbf{u})$ for each component $i \in \{1, \ldots, M\}$ is a contraction with respect to the block-maximum 2-norm.

The mapping $T(\mathbf{u})$ will become the mapping $R(\mathbf{u})$ if we choose $G_i = I^{n_{\mathbf{u}^i}}, \forall i$ (the identity matrix with the size $n_{\mathbf{u}^i}$), and take $F(\mathbf{u}) = \nabla_{\mathbf{u}} \tilde{\mathcal{L}}(\mathbf{u}, \mu^{(k)}) = 2\mathcal{H}\mathbf{u} + b$. With such choice, and evaluating the induced 2-norm in (4.86), the condition for contraction mapping of $R(\mathbf{u})$ is to find $\phi \in [0, 1)$ such that:

$$\|I^{n_{\mathbf{u}^i}} - 2\gamma \mathcal{H}_{ii}\|_2 + \sum_{j \neq i} \|2\gamma \mathcal{H}_{ij}\|_2 \leq \phi, \forall i \tag{4.87}$$

where $\mathcal{H}_{ij}$ with $i, j \in \{1, \ldots, M\}$ denotes the submatrix of $\mathcal{H}$ containing entries at rows belonging to subsystem i and columns belonging to subsystem j. Note that the matrix $I^{n_{\mathbf{u}^i}} - 2\gamma \mathcal{H}_{ii}$ inside the first induced matrix norm is a square, symmetric matrix, while the matrices $\mathcal{H}_{ij}$ are generally not symmetric, depending on the number of variables of each subsystem. The scalar $\phi \in [0, 1)$ is also the modulus of the contraction.

Using the properties of eigenvalue and singular value of matrices, we transform (4.87) into the following inequality:

$$\max_{\lambda} |2\gamma \lambda(\mathcal{H}_{ii}) - 1| + 2\gamma \sum_{j \neq i} \bar{\sigma}(\mathcal{H}_{ij}) \leq \phi, \forall i \tag{4.88}$$

where λ means eigenvalue, and $\bar{\sigma}$ denotes the maximum singular value.

In order to find $\gamma > 0$ and $\phi \in [0, 1)$ satisfying (4.88), we need:

$$\max_{\lambda} |2\gamma \lambda(\mathcal{H}_{ii}) - 1| + 2\gamma \sum_{j \neq i} \bar{\sigma}(\mathcal{H}_{ij}) < 1, \forall i \tag{4.89}$$

$$\Leftrightarrow \begin{cases} 2\gamma \lambda_{\max}(\mathcal{H}_{ii}) - 1 + 2\gamma \sum_{j \neq i} \bar{\sigma}(\mathcal{H}_{ij}) < 1 \\ 1 - 2\gamma \lambda_{\min}(H_{ii}) + 2\gamma \sum_{j \neq i} \bar{\sigma}(\mathcal{H}_{ij}) < 1 \end{cases}, \forall i \tag{4.90}$$

$$\Leftrightarrow \begin{cases} \gamma < 1/\left(\lambda_{\max}(\mathcal{H}_{ii}) + \sum_{j \neq i} \bar{\sigma}(\mathcal{H}_{ij})\right) \\ \lambda_{\min}(\mathcal{H}_{ii}) > \sum_{j \neq i} \bar{\sigma}(\mathcal{H}_{ij}) \end{cases}, \forall i \tag{4.91}$$

where $\lambda_{\max}(\cdot)$ and $\lambda_{\min}(\cdot)$ respectively denotes the maximum and the minimum eigenvalue of a square and symmetric matrix.

The first inequality of (4.91) shows how to choose γ, while the second inequality of (4.91) needs to be satisfied by the problem structure, which implies there are *weak dynamical couplings* between subsystems.

In summary, the mapping $R(\mathbf{u})$ satisfies (4.86) and thus is a contraction mapping if the following conditions hold:

1. For all i:

$$\lambda_{\min}(\mathcal{H}_{ii}) > \sum_{j \neq i} \bar{\sigma}(\mathcal{H}_{ij}) \tag{4.92}$$

2. The coefficient γ is chosen such that:

$$\gamma < \frac{1}{\lambda_{\max}(\mathcal{H}_{ii}) + \sum_{j \neq i} \bar{\sigma}(\mathcal{H}_{ij})}, \forall i \tag{4.93}$$

So, when condition (4.92) is satisfied and with γ chosen by (4.93), we can define $\phi \in (0, 1)$ as:

$$\phi = \max_i \left\{ \max \left\{ 2\gamma\big(\lambda_{\max}(\mathcal{H}_{ii}) + \sum_{j \neq i} \bar{\sigma}(\mathcal{H}_{ij})\big) - 1, \right. \right. \\ \left. \left. 1 - 2\gamma\big(\lambda_{\min}(\mathcal{H}_{ii}) - \sum_{j \neq i} \bar{\sigma}(\mathcal{H}_{ij})\big) \right\} \right\} \tag{4.94}$$

This ϕ is the modulus of the contraction $R(\mathbf{u})$, and also acts as the coefficient of the linear convergence rate of the Jacobi iteration (4.67), which means:

$$\|\mathbf{u}(p) - \mathbf{u}^*\|_{\mathrm{b-m},2,\mathfrak{p}} \leq \phi^p \|\mathbf{u}(0) - \mathbf{u}^*\|_{\mathrm{b-m},2,\mathfrak{p}}, \quad \forall p \geq 1 \tag{4.95}$$

where $\mathbf{u}^* = \arg\min_{\mathbf{u} \in \mathbf{\Omega}} \tilde{\mathcal{L}}(\mathbf{u}, x_t)$.

Note that the closer ϕ is to 0, the faster the aggregate update $\mathbf{u}(p)$ converges to the optimizer of the Lagrangian function.

In order to get the convergence rate w.r.t. the Euclidean norm, we will need a link from the Euclidean norm to the block-maximum norm:

$$\|x\|_2 \leq \sum_{i=1}^{M} \|x^i\|_2 \leq M \max_i \|x^i\|_2 = M \|x\|_{\mathrm{b-m},2,\mathfrak{p}} \tag{4.96}$$

Hence, the convergence rate of Jacobi iteration (4.67) w.r.t. the Euclidean norm is:

$$\|\mathbf{u}(p) - \mathbf{u}^*\|_2 \leq M \phi^p \max_i \|\mathbf{u}^i(0) - \mathbf{u}^{i*}\|_2, \quad \forall p \geq 1 \tag{4.97}$$

□

Remark 4.10 *Proposition 4.6 establishes a linear convergence rate of the Jacobi iteration, under the condition of* weak dynamical couplings *between subsystems. For the sake of illustrating condition (4.84), let all subsystems have the same number of inputs. Consequently, $\mathcal{H}_{ij}$ is a square and symmetric matrix for each pair (i, j). Hence, the maximum singular value $\bar{\sigma}(\mathcal{H}_{ij})$ equals to the maximum eigenvalue. Inequality (4.84) thus reads:*

$$\lambda_{\min}(\mathcal{H}_{ii}) > \sum_{j \neq i} \lambda_{\max}(\mathcal{H}_{ij}), \forall i$$

which implies that the couplings represented by $\mathcal{H}$ are small in comparison with each local cost.

Remark 4.11 *Note that the strong convexity of $\tilde{\mathcal{L}}$ and the condition (4.84) are required only for the convergence rate result of the Jacobi iteration in which $\tilde{\mathcal{L}}$ is a quadratic function. Extensions to other types of optimization problems, where the Lagrangian can be solved with bounded suboptimality, are immediate. In such cases we simply need to replace the Jacobi iteration with the appropriate algorithm that meets the suboptimality requirement in the inner loop, while the outer loop will remain intact.*

4.4.5 Determining the number of stopping iteration $\tilde{p}_k$ for the Jacobi algorithm

As $\tilde{\mathcal{L}}(\mathbf{u}, \cdot)$ is continuously differentiable in a closed bounded set $\mathbf{\Omega}$, it is Lipschitz continuous.

Suppose we know the Lipschitz constant Λ of $\tilde{\mathcal{L}}(\mathbf{u}, \cdot)$ over $\mathbf{\Omega}$, i.e., for any $\mathbf{u}^1, \mathbf{u}^2 \in \mathbf{\Omega}$ the following inequality holds:

$$\|\tilde{\mathcal{L}}(\mathbf{u}^1, \mu^{(k)}) - \tilde{\mathcal{L}}(\mathbf{u}^2, \mu^{(k)})\|_2 \leq \Lambda \|\mathbf{u}^1 - \mathbf{u}^2\|_2 \tag{4.98}$$

Taking $\mathbf{u}^1 = \mathbf{u}(\tilde{p}_k)$ and $\mathbf{u}^2 = \mathbf{u}^*$ in (4.98), and combining it with (4.85), we obtain:

$$\begin{aligned} \|\tilde{\mathcal{L}}(\mathbf{u}(\tilde{p}_k), \mu^{(k)}) - \min_{\mathbf{u} \in \mathbf{\Omega}} \mathcal{L}(\mathbf{u}, \mu^{(k)})\|_2 &\leq \Lambda \|\mathbf{u}(\bar{p}_k) - \mathbf{u}^*\|_2 \\ &\leq \Lambda M \phi^{\bar{p}_k} \max_i \|\mathbf{u}^i(0) - \mathbf{u}^{i*}\|_2 \end{aligned} \tag{4.99}$$

For each $i \in \{1, \ldots, M\}$, let D_i denote the diameter of the set $\mathbf{\Omega}_i$ w.r.t. the Euclidean norm, so we have $\|\mathbf{u}^i(0) - \mathbf{u}^{i*}\|_2 \leq D_i$. Hence the relation (4.99) can be further simplified as

$$\tilde{\mathcal{L}}(\mathbf{u}(\tilde{p}_k), \mu^{(k)}) \leq \min_{\mathbf{u} \in \mathbf{\Omega}} \tilde{\mathcal{L}}(\mathbf{u}, \mu^{(k)}) + \Lambda M \phi^{\tilde{p}_k} \max_i D_i \tag{4.100}$$

Based on (4.100), in order to use $\mathbf{u}(\tilde{p}_k)$ as the solution $\mathbf{u}^{(k)}$ that satisfies (4.65), we need an integer $\tilde{p}_k$ such that $\Lambda M \phi^{\tilde{p}_k} \max_i D_i \leq \varepsilon_t$, thus we choose:

$$\tilde{p}_k \geq \log_\phi \frac{\varepsilon_t}{\Lambda M \max_i D_i} \tag{4.101}$$

4.4.6 Properties of the HPF-DAG algorithm

The algorithm HPF-DAG has similar properties as the algorithm HPF-DEG, which are the existence of a feasible primal solution and the monotonic decrease of the cost function. Moreover, the algorithm HPF-DAG is built upon a distributed iterative algorithm, and hence almost all the computations are done by the local controllers, thus the only role of the coordinator is to define common parameters that will be transfered to local controllers at the beginning of each MPC sampling step.

Distributed Jacobi algorithm with guaranteed convergence

The computations in the inner loop can be executed in parallel by the subsystems. Let us define an r-step extended neighborhood of a subsystem i, denoted by $\mathcal{N}_r^i$, as the set containing all subsystems that can influence subsystem i within r successive time steps. So $\mathcal{N}_r^i$ is the union of subsystem indices in the neighborhoods of all subsystems in $\mathcal{N}_{r-1}^i$:

$$\mathcal{N}_r^i = \bigcup_{j \in \mathcal{N}_{r-1}^i} \mathcal{N}^j \tag{4.102}$$

where $\mathcal{N}_1^i = \mathcal{N}^i$. We can see that in order to get update information in the Jacobi iterations, each subsystem i needs to communicate only with subsystems in $\mathcal{N}_{N-1}^i$, where N is the prediction horizon. This set includes all other subsystems that couple with i in the problem (4.10)–(4.12) after eliminating the state variables. This communication requirement indicates that we will benefit from communication reduction when the number of subsystems M is much larger than the horizon N, and the coupling structure is sparse.

Assume that the weak coupling condition (4.84) holds. Then after $\tilde{p}_k$ iterations as computed by (4.101), the Jacobi algorithm generates a solution $\mathbf{u}^{(k)} \triangleq \mathbf{u}(\tilde{p}_k)$ that satisfies (4.65) in the outer loop.

Feasible primal solution

Proposition 4.7 *Suppose Assumptions 4.1–4.5 hold. Construct g' as in (4.27), $\tilde{\alpha}_t$ as in (4.43). Let the outer loop (4.65)–(4.66) with $\mu^{(0)} = 0 \cdot \mathbf{1}_m$ be iterated for $k = 0, \ldots, \tilde{k}_t$. Then $\hat{\mathbf{u}}^{(\tilde{k}_t)}$ is a feasible solution of (4.10)–(4.12), where $\hat{\mathbf{u}}^{(\tilde{k}_t)}$ is the primal average, computed by (4.45).*

Proof. We now make use of Proposition 4.5. With a finite number of $\tilde{k}_t$ iterations (4.68) reads as

$$\left\| \left[g'\left(\hat{\mathbf{u}}^{(\tilde{k}_t)}, x_t \right) \right]^+ \right\|_2 \leq \frac{1}{\tilde{k}_t \tilde{\alpha}_t} \left(\frac{3}{\gamma_t} \left[f(\bar{\mathbf{u}}_t, x_t) - \tilde{q'}_t^* \right] + \frac{\tilde{\alpha}_t {L'_t}^2}{2\gamma_t} + \tilde{\alpha}_t L'_t \right) \tag{4.103}$$

Moreover, the dual function $\tilde{q'}_t$ is a concave function, and therefore $\tilde{q'}_t^* \geq \tilde{q'}(0, x_t)$. Recall that $f(\mathbf{u}, x_t) > 0, \forall \mathbf{u} \neq 0, x_t \neq 0$. Thus, $\tilde{q'}(0, x_t) = \min_{\mathbf{u} \in \mathbf{\Omega}} f(\mathbf{u}, x_t) + 0 \cdot \mathbf{1}_m^T g'(\mathbf{u}, x_t) =$

$\min_{\mathbf{u}\in\mathbf{\Omega}} f(\mathbf{u}, x_t) > 0$, and therefore

$$\left\| \left[g'\left(\hat{\mathbf{u}}^{(\tilde{k}_t)}, x_t\right) \right]^+ \right\|_2 < \frac{1}{\tilde{k}_t \tilde{\alpha}_t} \left(\frac{3}{\gamma_t} f(\bar{\mathbf{u}}_t, x_t) + \frac{\tilde{\alpha}_t L_t'^2}{2\gamma_t} + \tilde{\alpha}_t L_t' \right) \tag{4.104}$$

Combining (4.104) with (4.81), and noticing that $\tilde{k}_t$ and $\tilde{c}_t$ are all positive leads to

$$\left\| \left[g'\left(\hat{\mathbf{u}}^{(\tilde{k}_t)}, x_t\right) \right]^+ \right\|_2 < \tilde{c}_t \tag{4.105}$$

$$\Rightarrow g_j'\left(\hat{\mathbf{u}}^{(\tilde{k}_t)}, x_t\right) < \tilde{c}_t, \quad j = 1, \ldots, m \tag{4.106}$$

$$\Rightarrow g_j\left(\hat{\mathbf{u}}^{(\tilde{k}_t)}, x_t\right) < 0, \quad j = 1, \ldots, m \tag{4.107}$$

where the last inequality implies that $\hat{\mathbf{u}}^{(\tilde{k}_t)}$ is a feasible solution of problem (4.10)–(4.12), due to $\tilde{c}_t < \min_{j=1,\ldots,m}\{-g_j(\bar{\mathbf{u}}_t, x_t)\}$. □

Closed-loop stability

Proposition 4.8 *Suppose Assumptions 4.1–4.5 hold. Then the solution $\hat{\mathbf{u}}^{(\tilde{k}_t)}$ generated by Algorithm 4.4 satisfies the following inequality:*

$$f(\mathbf{u}_t, x_t) < f(\mathbf{u}_{t-1}, x_{t-1}), \quad \forall t \in \mathbb{Z}_+ \tag{4.108}$$

Proof. Using (4.53) and (4.78), and noting that $\mu^{(0)} = 0$, we obtain:

$$f\left(\hat{\mathbf{u}}^{(\tilde{k}_t)}, x_t\right) \leq \tilde{f}'^*_t + \frac{\|\mu^{(0)}\|}{2\tilde{k}_t\tilde{\alpha}_t} + \frac{\tilde{\alpha}_t L_t'^2}{2} + \varepsilon_t \leq \tilde{f}'^*_t + \Delta_t \tag{4.109}$$

Notice that $\bar{\mathbf{u}}_t$ is also a feasible solution of (4.62)–(4.64) (due to the way we construct the tightened problem: $\bar{\mathbf{u}}_t$ still belongs to the interior of the tightened constraint set), while $f_t'^*$ is the optimal cost value of this problem. As a consequence,

$$\tilde{f}'^*_t \leq f(\bar{\mathbf{u}}_t, x_t) \tag{4.110}$$

Combining (4.109), (4.110), and (4.14), and noting that $\mathbf{u}_t = \hat{\mathbf{u}}^{(\tilde{k}_t)}$ leads to:

$$f(\mathbf{u}_t, x_t) < f(\mathbf{u}_{t-1}, x_{t-1}), \quad \forall t \in \mathbb{Z}_+ \tag{4.111}$$

□

Note that besides the monotonic decrease of $f(\mathbf{u}_t, x_t)$, all the other conditions for Lyapunov stability of MPC (Mayne et al., 2000) are satisfied. Therefore, Proposition 4.8 leads to closed-loop MPC stability, where the cost function $f(\mathbf{u}_t, x_t)$ is a Lyapunov candidate function.

4.5 Realization of the assumptions

In this section, we discuss a method to update the Slater vector and the constraint norm bound for each time step, implying that Assumptions 4.3 and 4.5 are only necessary in the first time step ($t = 0$).

4.5.1 Updating the Slater vector

Lemma 4.9 *Suppose Assumption 4.2 holds. Let* $\mathbf{u}_t$ *be the solution of the MPC problem (4.3)–(4.9) at time step t, computed by Algorithm 4.4. Then* $\tilde{\mathbf{u}}_{t+1}$ *constructed by shifting* $\mathbf{u}_t$ *one step ahead and adding* $\tilde{u}_{t+N} = Kx_{t+N}$*, is a Slater vector for constraint (4.11) at time step* $t + 1$.

Proof. Note that $\hat{\mathbf{u}}^{(\bar{k}_t)}$ is a feasible solution of problem (4.10)–(4.12). Moreover, the strict inequality (4.107) means that $\hat{\mathbf{u}}^{(\bar{k}_t)}$ is in the interior of the constraint set of (4.3)–(4.9). This also yields:

$$x_{t+N} \in \mathrm{int}(X_\mathrm{f}) \tag{4.112}$$

Moreover, due to Assumption 4.2, we have $(A + BK)x_{t+N} \in \mathrm{int}(X_\mathrm{f})$. This means that if we use $\tilde{u}_{t+N} = Kx_{t+N}$, then the next state is also in the interior of the terminal constraint set X_f. Note that $\mathcal{U}$ and $\mathcal{X}$ do not change when problem (4.3)–(4.9) is shifted from t to $t + 1$. Hence, all the inputs of $\tilde{\mathbf{u}}_{t+1}$ and their subsequent states are in the interior of the corresponding constraint sets. Therefore, $\tilde{\mathbf{u}}_{t+1}$ as constructed at step 5 of Algorithm 4.4 is a Slater vector for the constraint (4.11) at time step $t + 1$. □

This means we can use $\bar{\mathbf{u}}_{t+1} = \tilde{\mathbf{u}}_{t+1}$ as the qualifying Slater vector for Assumption 4.3 at time step $t + 1$.

4.5.2 Updating the constraint norm bound

In our general problem setup, $g(\mathbf{u}, x)$ is composed of affine functions over $\mathbf{u}$ and x, and thus can be written compactly as

$$g(\mathbf{u}, x) = \Xi x + \Theta \mathbf{u} + \tau \tag{4.113}$$

with constant matrices Ξ, Θ and a constant vector τ. Then for each x_{t-1}, x_t, and $\mathbf{u} \in \mathbf{\Omega}$, the following holds:

$$\begin{aligned} &g(\mathbf{u}, x_t) = g(\mathbf{u}, x_{t-1}) + \Xi(x_t - x_{t-1}) \\ &\Rightarrow \|g(\mathbf{u}, x_t)\|_2 \leq \|g(\mathbf{u}, x_{t-1})\|_2 + \|\Xi(x_t - x_{t-1})\|_2 \end{aligned} \tag{4.114}$$

In order to find a bound L_t for $g(\mathbf{u}, x_t)$ in each step t with $t \geq 1$, we assume to have the constraint norm bound available from the previous step:

$$L_{t-1} \geq \|g(\mathbf{u}, x_{t-1})\|_2, \forall \mathbf{u} \in \mathbf{\Omega} \tag{4.115}$$

Combining the above inequalities a norm bound update for $g(\mathbf{u}, x_t)$ can be obtained as:

$$L_t = L_{t-1} + \|\Xi(x_t - x_{t-1})\|_2 \tag{4.116}$$

Note that if we do not want to update L_t at every step t, there is an alternative method to compute a bound $L_{\max}$ that fits all steps. By formulating a nonlinear maximization problem with the cost function is the norm of $g(\mathbf{u}, x_t)$, where both $\mathbf{u} \in \mathbf{\Omega}$ and $x_t \in \mathcal{X}$ are the variables, we can solve this problem offline and assign the result to $L_{\max}$. This value $L_{\max}$ is also the upper limit for L_t that is updated by (4.116), hence L_t will not grow too high that could lead to inefficient estimation of $\tilde{k}_t$.

Remark 4.12 *When Algorithm 4.1 is used to solve the exact dual problem, we need to use L_t as the norm bound for the function $h(\mathbf{u}, x_t)$. The similar way for updating L_t can be used, by replacing $g(\mathbf{u}, x_t)$ by $h(\mathbf{u}, x_t)$.*

4.6 Simulation example

In this section, we present the results of an application example on MPC with the constraint tightening approach proposed in this chapter. We focus on applying the Algorithm HPF-DAG, as it is less conservative than the Algorithm HPF-DEG, i.e., the interior of the constraint set $g(\mathbf{u}, x_0) \geq 0$ is larger than the interior of the set $h(\mathbf{u}, x_0) \geq 0$, moreover Algorithm HPF-DAG allows the control inputs to reach the hard limits, while with Algorithm HPF-DEG the bounds for control inputs are also tightened, leading to a reduction of MPC performance.

The example system is the same canal system that is described in Section 2.4, with the same control problem. There are some differences in the setting used in this chapter in comparison to the example used in Chapter 2: here the terminal point constraint $x_N = 0$ is removed, the state variables are eliminated from the MPC optimization problem by using dynamical constraints, the prediction horizon is $N = 5$ (a shorter horizon so that the number of inequality constraints to be tightened is less).

The Algorithm HPF-DAG is used in this example, it is expected to generate at every MPC step t a feasible solution with respect to the physical constraints:

$$x_{\min} \leq x_{t+k} \leq x_{\max},\ k = 1, \ldots, N \tag{4.117}$$

$$u_{\min} \leq u_{t+k} \leq u_{\max},\ k = 0, \ldots, N-1 \tag{4.118}$$

The constraints described in (4.117) are then packed into the inequality constraint $g(\mathbf{u}, x_t) \geq 0$ after the state variables are eliminated, and the constraints (4.118) are cast into the form $\mathbf{u} \in \mathbf{\Omega}$ in this chapter.

With $x_{\min} = -0.5I_4, x_{\max} = 0.5I_4, u_{\min} = -0.5I_5, u_{\max} = 0.5I_5$, and an initial nonzero state x_0, we are able to find a Slater vector such that $\tilde{c}_0 = 0.2$ (recall $\tilde{c}_t$ in (4.28)), and the initial constraint norm bound $L_0 = 13.4$.

The control example is simulated in 20 MPC steps. In each MPC step t, Algorithm 4.4 is used to generate a solution to the MPC optimization problem (4.3)–(4.9), with the number

of outer loop iterations $\tilde{k}_t$ determined by (4.81), and the number of inner loop iterations $\tilde{p}_k$ is determined by (4.101). Then the first control input in the solution generated by Algorithm 4.4 is used to simulate the system, and the routine restarts with the next MPC step. In every MPC step, we also solve the centralized optimization problem to get the optimal solution and the optimal cost, for a comparison purpose.

The simulation results give feasible solution at each MPC step. For illustration, the predicted input generated by Algorithm 4.4 in the first MPC step is compared with the optimal solution and the Slater vector, given in Figure 4.1. These inputs are all feasible, recall that the constraint set for every input is $[-0.5; 0.5]$.

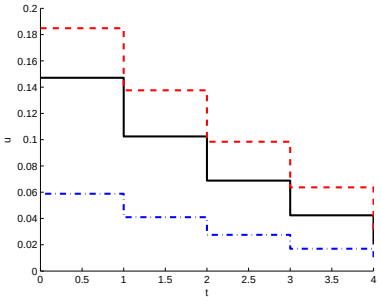
In Figure 4.2(a), the number of outer loop iterations $\tilde{k}_t$ and the average number of inner loop iterations $\tilde{p}_k$ for all $k \in \{1, \ldots, \tilde{k}_t\}$, denoted by $\tilde{p}_{\max_t}$, are plotted for each sampling step. The corresponding CPU time spent by Algorithm 4.4 at each sampling step is showed in Figure 4.2(b). The CPU time is measured by implementing all the computations in one PC, running MATLAB on Windows with an Intel(R) Core(TM) i7 CPU at 2.30 GHz and with 8 GB RAM. We can see that the simulation takes long time for computation, since at every sampling step, there must be $\tilde{k}_t \times \tilde{p}_{\max_t} \times M$ small optimization problems to be solved, where M is the number of subsystems (in this example $M = 4$). If the algorithm is simulated in a distributed setting, the computation task would be divided into M subsystems, thus the computation time is much less than the total number we obtain in this simulation.

Note that the computation time is amplified by the number of iterations required to minimize the Lagrangian function using the distributed Jacobi algorithm, which can be considered as a price for using a very straightforward method to solve a centralized optimization problem. As shown in Proposition 4.6, the convergence rate of the distributed Jacobi algorithm is influenced by the coupling nature of the problem, i.e., depending on the level of coupling presented by the Hessian of the problem, we can obtain a bigger (worse) or smaller (better) value of ϕ - the contraction modulus that has a key role for the convergence of the distributed Jacobi algorithm.

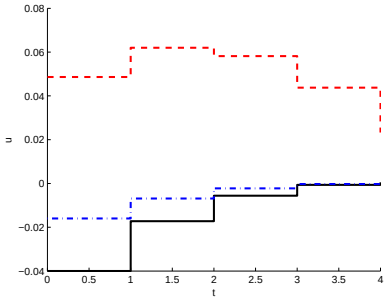
The evolutions of cost functions associated with the MPC solutions are plotted in Figure 4.2. In this figure, we compare the optimal cost, the cost associated with the Slater vector, the cost generated by Algorithm 4.4, and the upper bound of the cost for guaranteeing stability. Although it is easy to obtain closed-loop stability with this example, the comparison of the cost nevertheless confirms that the upper bound of the cost is always respected by the averaging result of Algorithm 4.4. Moreover, as t increases, the upper bound is put closer to the optimal cost, thus the performance of the Algorithm HPF-DAG is also closer to the optimal centralized MPC performance.

4.7 Conclusions

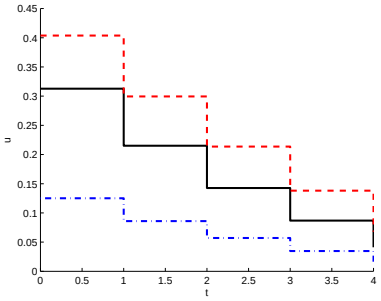
In this chapter, we have presented a constraint tightening approach for solving MPC optimization problems involving large-scale systems with coupling in dynamics and constraints. This new approach provides guaranteed feasibility and stability after a finite number of iterations. We have presented two variations following this approach: the HPF-DEG algorithm is based on conjugate gradient method, and the HPF-DAG one is based on a



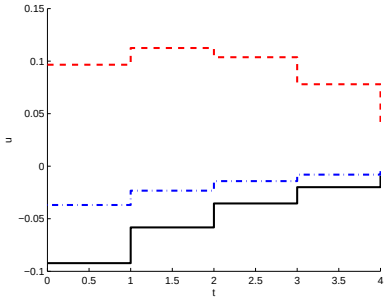
(a) Prediction of u_1



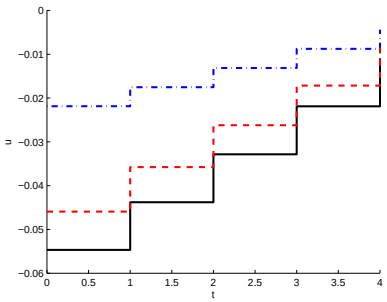
(b) Prediction of u_2



(c) Prediction of u_3

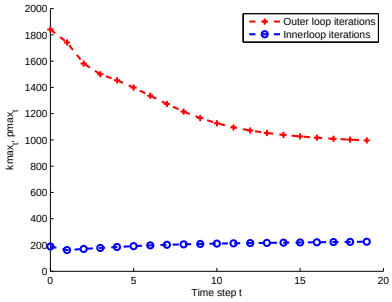


(d) Prediction of u_4

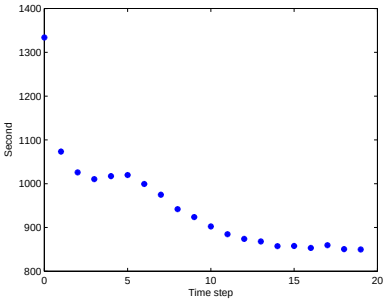


(e) Prediction of u_5

Figure 4.1: *Comparison of predicted input series in the first sampling step, between the optimal solution (black), the initial Slater vector (dash-dotted blue), and the feasible solution of Algorithm 4.4 (dashed red).*



(a) Numbers of iterations of Algorithm 4.4 at each MPC step



(b) Computation time for the simulation of Algorithm 4.4 at each MPC step

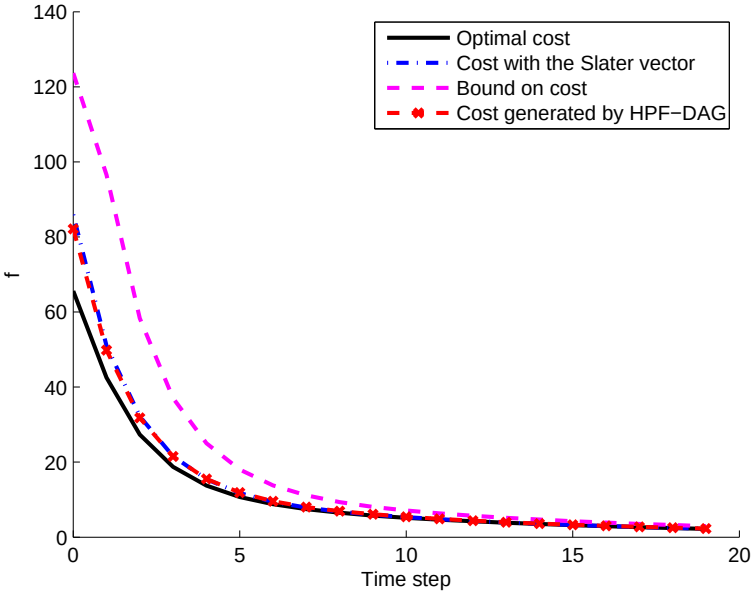


Figure 4.2: *Comparison of evolutions of cost functions during MPC simulation*

Jacobi iterative method, and both variations combine this with a simple projected gradient update for the dual problem. These algorithms facilitate implementation in a hierarchical way. The performance of this approach is discussed with an illustrative example of a simple canal system. Future extensions of this approach includes *a posteriori* choice of the solution between either the Slater vector $\bar{\mathbf{u}}_t$ or the primal average $\hat{\mathbf{u}}^{(\bar{k}_t)}/\hat{\mathbf{u}}^{(\tilde{k}_t)}$ generated by the algorithm HPF-DEG/HPF-DAG. Another topic for research is to find the optimal parameters $\alpha_t/\tilde{\alpha}_t$, $c_t/\tilde{c}_t$, and ε_t so that the algorithms take the least amount of time to generate a feasible solution that leads to monotonic decrease of the MPC cost function.

Chapter 5

Application of distributed model predictive control to a hydro power valley

In this chapter, we design a distributed model predictive controller for the power-reference tracking problem of a Hydro Power Valley (HPV) system, using the distributed accelerated proximal gradient method that was described in Chapter 3. There are a number of challenges posed by the HPV benchmark, including the nonlinearity, nonsmoothness, and couplings in the constraints and in the cost function. We propose approximation methods for modeling of the HPV, and approximate optimization problems that fits in the DMPC framework. Through simulations, DMPC is shown to be able to achieve comparable performance as centralized MPC, while using neighbor-to-neighbor communication only. Provided numerical studies also suggest that for the sparsely interconnected system at hand, the proposed distributed solver is more computationally efficient than a standard centralized QP solver.

5.1 Introduction

Hydro power plants generate electricity from potential energy and kinetic energy of the natural water, and often a number of power plants are placed along a long river or water body systems to generate the power at different stages. Currently, hydro power is the most important means of renewable power generation in the world. In order to meet the world's electricity demand, hydro power production should continue to grow due to the increasing cost of fossil fuels. However, hydroelectricity, like any renewable energy, depends on the availability of a primary resource, in this case: water. Most natural locations, where power-generating infrastructure can be built economically, have already been utilized (PEW Center on Global Climate Change, 2011). The expected trend for future use of hydro-power is to build small-scale plants that can generate electricity for a single community. Thus,

an increasingly important objective of hydro power plants is to manage the available water resources efficiently, while following an optimal production profile with respect to the change in the electricity market, to maximize the long-term benefit of the plant. This water resource management must be compatible with ship navigation and irrigation, and it must respect environmental and safety constraints on levels and flow rates in the lakes and the rivers. This is why the real-time control of the water flows in a Hydro Power Valley (HPV) becomes important and can increase significantly the power efficiency of these systems.

An HPV may contain several rivers and lakes, spanning a wide geographical area and exhibiting complex dynamics. In order to tackle the plant-wide control of such a complex system, an HPV is often treated as a large-scale system consisting of interacting subsystems. Large-scale system control has been an active research area that has resulted in a variety of control techniques, which can be characterized by several main categories: decentralized control, distributed control, as well as centralized control. Such control approaches can be found in a rich literature on control of water canals for irrigation and hydrosystems (Mareels et al., 2005, Litrico and Fromion, 2009). For the control problem of open water systems, centralized MPC has been studied with simulations of nonlinear MPC in combination with model smoothing and/or model reduction techniques (Igreja and Lemos, 2009, Nederkoorn et al., 2011), and with linear MPC of low-dimension systems in real implementations (van Overloop, 2006, van Overloop et al., 2010). However, centralized MPC has a drawback when controlling large-scale systems due to limitations in communications and the computational burden. These issues fostered the studies of decentralized MPC and distributed MPC for large-scale water systems. Early decentralized MPC methods for irrigation canals used the decomposition-coordination approach to obtain decentralized versions of LQ control (Fawal et al., 1998). Several decentralized MPC simulations applied to irrigation canals and rivers were presented in (Georges, 1994, Sawadogo et al., 1998, Gomez et al., 2002, Sahin and Morari, 2010). Distributed MPC approaches based on coordination and cooperation for water delivery canals were presented in Georges (1994), Negenborn et al. (2009), Igreja et al. (2011), Anand et al. (2011). The typical control objective in these studies is to regulate water levels and to deliver the required amount of water to the right place at some time in the future, i.e., the cost function does not have any special term except the penalties on the states and the inputs. On the other hand, in hydro power control, there are output penalty terms in the cost function that represent the objective of manipulating power production. Recent literature taking into account this cost function includes centralized nonlinear MPC with a parallel version of the multiple-shooting method for the optimal problem using continuous nonlinear dynamics (Savorgnan et al., 2011), and a software framework that formulates a discrete-time linear MPC controller with the possibility to integrate a nonlinear prediction model and to use commercial solvers to solve the optimization problem (Petrone, 2010). The hydro power control problem considered in this chapter is similar to the set-up in Savorgnan et al. (2011), Petrone (2010). However, it distinguishes itself by using a distributed control structure that aims to avoid global communications and to divide the computational tasks into local sub-tasks that are handled by subsystems, making the approach more suitable for scaling up to other even more complicated hydro power plants.

The proposed distributed MPC design approach is enabled by a state-of-the-art distributed optimization algorithm that has recently been developed in Chapter 3. This optimization algorithm is designed for a class of strongly convex problems with mixed 1-norm and 2-norm

terms in the cost function, which perfectly suits the power-reference tracking objective in HPV control. The underlying optimization algorithm, although being implemented in a distributed way, is proved to achieve the global optimum with an $O(\frac{1}{k^2})$ convergence rate, which is a significant improvement compared to the distributed MPC methods presented in Chapter 2 and in Giselsson and Rantzer (2010), Negenborn et al. (2008), which achieve an $O(\frac{1}{k})$ convergence rate. By means of numerical examples, we will demonstrate the fast convergence property of the distributed algorithm, which can outperform a centralized QP approach for solving the same optimization problem.

The remaining parts of the chapter are organized as follows. In Section 5.2, we describe the hydro power valley (HPV) system and the power-reference tracking problem. Section 5.3 summarizes the treatment of the HPV model, followed by the control framework in Section 5.4, where we describe the distributed MPC algorithm that is used in the simulations. The simulation results are presented in Section 5.5 which also features a comparison with centralized MPC and decentralized MPC. Through the various aspects of the comparison including performance, computational efficiency, and communication requirements, the advantages of the distributed MPC algorithm will be highlighted. Section 5.6 concludes the chapter and outlines future work.

5.2 Problem description

In this section, we provide a summary to the HD-MPC hydro power valley benchmark (Savorgnan and Diehl, 2011) and then discuss the main control challenges.

5.2.1 Hydro power valley system

We consider a hydro power plant composed of several interconnected subsystems, as illustrated in Figure 5.1. The plant can be divided into 8 subsystems, of which subsystem S_1 is composed of the lakes L_1, L_2, the duct U_1 connecting them, and the ducts C_1, T_1 that connect L_1 with the reaches[1] R_1, R_2, respectively. Subsystem S_2 is composed of the lake L_3 and the ducts C_2, T_2 that connect L_3 to the reaches R_4, R_5, respectively. There are 6 other subsystems, each of which consists of a reach and the dam at the end of the reach. These six reaches R_1, R_2, R_3, R_4, R_5, and R_6 are connected in series, separated by the dams D_1, D_2, D_3, D_4, and D_5. The large lake that follows the dam D_6 is assumed to have a fixed water level, which will absorb all the discharge. The outside water flows enter the system at the upstream of the reach R_1 and at the middle of the reach R_3.

There are structures placed in the ducts and at the dams to control the flows. These are the turbines placed in the ducts T_1, T_2 and at each dam for power production. In the ducts C_1, C_2 there are composite structures that can either function as pumps (for transporting water to the lakes) or as turbines (when water is drained from the lakes).

The whole system has 10 manipulated variables, which are composed by six dam flows (q_{D1}, q_{D2}, q_{D3}, q_{D4}, q_{D5}, q_{D6}), two turbine flows (q_{T1}, q_{T2}), and two pump/turbine flows (q_{C1}, q_{C2}).

[1] A reach is a river segment between two dams.

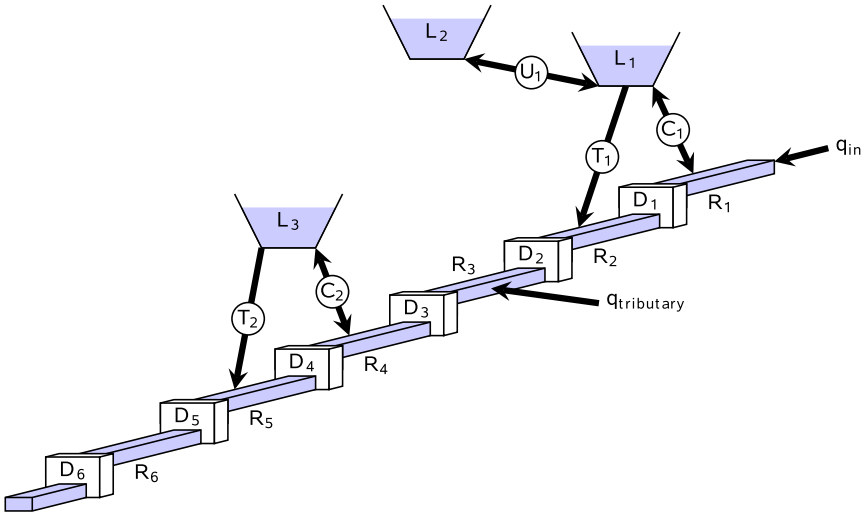


Figure 5.1: *Overview of the HD-MPC hydro power valley system (Savorgnan and Diehl, 2011)*

The detailed setup of the HPV is given in Savorgnan and Diehl (2011), with the continuous-time nonlinear model constructed by using the Saint Venant partial differential equations and employing spatial discretization for obtaining a system of ordinary differential equations.

5.2.2 Power-reference tracking problem

The control problem is to track a power production profile on a daily basis with the following cost function:

$$\begin{aligned} J \triangleq & \int_0^T \gamma \left| p_{\mathrm{r}}(t) - \sum_{i=1}^{8} p_i(x(t), u(t)) \right| dt \\ & + \sum_{i=1}^{8} \int_0^T (x_i(t) - x_{\mathrm{ss},i})^T Q_i (x_i(t) - x_{\mathrm{ss},i}) dt \\ & + \sum_{i=1}^{8} \int_0^T (q_i(t) - q_{\mathrm{ss},i})^T R_i (q_i(t) - q_{\mathrm{ss},i}) dt \end{aligned} \tag{5.1}$$

subject to the nonlinear dynamics and the operational constraints on water levels and water flows, which are denoted by the variables x and q, respectively. The index i is the subsystem index. The weights $Q_i, R_i, i = 1, \ldots, 8$, γ, and the testing period T are parameters of the benchmark.

Note that the cost function (5.1) contains two components. The power tracking error is put in the 1-norm term to reflect the economical cost, with p_{r} the power reference and p_i the

power produced/consumed by subsystem $i \in \{1, \ldots, 8\}$. The quadratic term in the cost function represents the penalties on the state deviation from the steady state x_{ss} and the energy used for manipulating the inputs away from steady state flows q_{ss}.

The function p_i for $i = 1, 2$ is the produced/consumed power (c.f., Savorgnan and Diehl (2011)):

$$p_i(x(t), q(t)) = k_{C_i}(q_{C_i}(t))q_{C_i}(t)\Delta x_{C_i}(t) + k_{T_i}q_{T_i}(t)\Delta x_{T_i}(t) \tag{5.2}$$

where q_{C_i} and q_{T_i} are flows through ducts C_i and T_i, Δx_{C_i} and Δx_{T_i} are the relative differences in water levels before and after ducts C_i and T_i respectively, k_{T_i} is the turbine power coefficient and

$$k_{C_i}(q_{C_i}(t)) = \begin{cases} k_{T_{C_i}} & q_{C_i}(t) \geq 0 \\ k_{P_{C_i}} & q_{C_i}(t) < 0 \end{cases}$$

is a discontinuous power coefficient that depends on whether duct C_i acts as a pump or a turbine. For $i = 3, \ldots, 8$ we have

$$p_i(x(t), q(t)) = k_{D_{i-2}}q_{D_{i-2}}(t)\Delta x_{D_{i-2}}(t) \tag{5.3}$$

which is the power produced by the turbine located at dam D_{i-2}.

Remark 5.1 *There are several challenges posed by this control problem, of which the most critical ones are the relatively large size, the nonlinearity and moreover nonsmoothness (for subsystems 1 and 2) of produced/consumed power functions. In the current setting, the rate for updating control input is not critical, with* $T_{\mathrm{s}} = 30\mathrm{min}$ *that is long enough for solving a centralized MPC problem online. For demonstration, we will apply a distributed accelerated proximal gradient algorithm (cf. Chapter 3) for the hydro power valley control problem to show that distributed MPC can achieve close performance to centralized MPC, while avoiding the communication need and being even more computationally efficient than a classical centralized MPC method.*

Before describing the distributed MPC approach, we will first discuss the modeling of the hydro power valley in the next section.

5.3 Hydro power valley modeling

5.3.1 Nonlinear modeling, spatial discretization and linearization

The model of the reaches is based on the one-dimensional Saint Venant partial differential equation, representing the mass and momentum balance:

$$\begin{cases} \dfrac{\partial q(t,z)}{\partial z} + \dfrac{\partial s(t,z)}{\partial t} = 0 \\ \dfrac{1}{g}\dfrac{\partial}{\partial t}\left(\dfrac{q(t,z)}{s(t,z)}\right) + \dfrac{1}{2g}\dfrac{\partial}{\partial z}\left(\dfrac{q^2(t,z)}{s^2(t,z)}\right) + \dfrac{\partial h(t,z)}{\partial z} + I_{\mathrm{f}}(t,z) - I_0(z) = 0 \end{cases} \tag{5.4}$$

with z the spatial variable, t the time variable, q the river flow (or discharge), s the cross-section surface of the river, h the water level w.r.t. the river bed, I_{f} the friction slope, $I_0(z)$ the river bed slope, and g the gravitational acceleration constant.

The partial differential equation (5.4) can be converted into a system of ordinary differential equations by using spatial discretization. To this aim, each reach is divided into 20 cells, yielding 20 additional states, which are the water levels at the beginning of the cells. As a consequence, a set of nonlinear dynamical equations is obtained, with in total 249 states (Savorgnan and Diehl, 2011).

A linear model of the hydro power valley is obtained by linearizing the nonlinear model at a steady operating condition, followed by discretization in time. Due to the coupling structure of the system, the discrete-time linear model of each of the eight subsystems $i = 1, \ldots, 8$ can be expressed in the following form:

$$\begin{aligned} x_i(k+1) &= A_{ii}x_i(k) + \sum_{j=1}^{8} B_{ij}u_j(k) \\ y_i(k) &= C_i x_i(k) \end{aligned} \tag{5.5}$$

in which the variables x, u, and y stand for the deviation from the steady-state values, and the subscripts i, j stand for the subsystem index. Note that the subsystems are coupled through the inputs only.

The use of a discrete-time linearized model enables controller design with some specific approaches, which include the distributed accelerated proximal gradient (DAPG) algorithm that has been developed in Chapter 3 of this thesis. However, the linear discrete-time model cannot be directly used in an MPC context due to the existence of a number of uncontrollable and unobservable modes. These uncontrollable/unobservable modes are a result of the discretization in time since we cannot expect to control/observe the water levels independently in each segment in every reach due to the dependencies of the water levels in adjacent segments. Moreover, the linear model has a large number of states, causing computational burden. Therefore, we will use balanced truncation for model order reduction (Moore, 1981, Gugercin and Antoulas, 2004) to remove less significant modes, including all unobservable and uncontrollable ones.

5.3.2 Decentralized model order reduction

The block-diagonal structure of the discrete-time dynamical system (5.5) makes it possible to do model reduction on each subsystem individually. Under the condition that the model (5.5) is stable, reachable and observable (which can be easily verified), we can use balanced truncation (Gugercin and Antoulas, 2004) to reduce the order of each local model (5.5). Below we briefly review the balanced truncation technique. To this end we introduce $B_i = [B_{i1} \ldots B_{i8}]$ and $u = [(u_1)^T \ldots (u_8)^T]^T$ to get the following discrete-time linear model of each subsystem:

$$\begin{aligned} x_i(k+1) &= A_{ii}x_i(k) + B_i u(k) \\ y_i(k) &= C_i x_i(k) \end{aligned} \tag{5.6}$$

We can compute the local controllability Gramian $W_{\mathrm{c}i}$ by solving the following matrix equation:

$$A_{ii}W_{\mathrm{c}i}A_{ii}^T + B_iB_i^T = W_{\mathrm{c}i} \tag{5.7}$$

and the local observability Gramian $W_{\mathrm{o}i}$ by:

$$A_{ii}^TW_{\mathrm{o}i}A_{ii} + C_i^TC_i = W_{\mathrm{o}i} \tag{5.8}$$

Let us consider a state transformation $\bar{x}_i = Tx_i$ with an invertible matrix T. The Gramians are transformed to:

$$\bar{W}_{\mathrm{c}_i} = TW_{\mathrm{c}i}T^T, \quad \bar{W}_{\mathrm{o}_i} = T^{-T}W_{\mathrm{o}i}T^{-1} \tag{5.9}$$

For each subsystem i, we can find a particular matrix T_i such that

$$\bar{W}_{\mathrm{c}_i} = \bar{W}_{\mathrm{o}_i} = \mathrm{diag}(\sigma_{1i}, \ldots, \sigma_{ni}) \tag{5.10}$$

with $\sigma_{j_i} \geq 0, \forall j$ and $\sigma_{1i} \geq \sigma_{2i} \geq \ldots \geq \sigma_{ni}$. This is called a balancing transformation (Moore, 1981). The controllability and observability Gramians of the new system realization are equal and diagonal, consisting of entries $\sigma_{1i}, \ldots, \sigma_{ni}$ which are called Hankel singular values.

The truncated model is obtained by removing the modes that correspond to small σ_{j_i}. All the modes of the reduced model are both controllable and observable (corresponding to $\sigma_{j_i} > 0$). Denoting the number of kept modes by n_i^{r} for each subsystem i the resulting transformation matrices become

$$T_i^{\mathrm{r}} = \begin{bmatrix} T_{i[1\bullet]} \\ \vdots \\ T_{i[n_i^{\mathrm{r}}\bullet]} \end{bmatrix} \qquad T_i^{\mathrm{r,inv}} = [T^{-1}_{i[\bullet 1]} \ldots T^{-1}_{i[\bullet n_i^{\mathrm{r}}]}] \tag{5.11}$$

where $T_{i[j\bullet]}$ denotes the j-th row of T_i and $T^{-1}_{i[\bullet j]}$ denotes the j-th column of T_i^{-1}. By denoting the new state variables, $x_i^{\mathrm{r}} = T_i^r x_i^{\mathrm{d}}$, and the control variable $q^{\mathrm{r}} = q$, we represent the reduced order model as:

$$x_i^{\mathrm{r}}(k+1) = A_{ii}^{\mathrm{r}}x_i^{\mathrm{r}}(k) + B_i^{\mathrm{r}}q^{\mathrm{r}}(k) \tag{5.12}$$

$$y_i^{\mathrm{r}}(k) = C_i^{\mathrm{r}}x_i^{\mathrm{r}}(k) \tag{5.13}$$

where $A_{ii}^{\mathrm{r}} = T_i^{\mathrm{r}}A_{ii}T_i^{\mathrm{r,inv}}$, $B_i^{\mathrm{r}} = T_i^{\mathrm{r}}B_i$ and $C_i^{\mathrm{r}} = C_iT_i^{\mathrm{r,inv}}$. It should be noted that the sparsity structure of B_i^{r} is the same as in the non-reduced input matrix B_i.

This balanced truncation is performed for each local subsystem. The aggregate truncated model can then easily be constructed. By truncating less important modes which are sorted based on the relative proportion of the Hankel singular values, we can obtain a 32-state reduced model that approximately represents the dynamics of the full linear model with 249 states. As now we have a small-size system, real-time control could be fulfilled relatively easily by centralized MPC, however here we focus on illustrating benefits of distributed MPC on other aspects, notably performance comparison with centralized MPC with the

difference in communication requirement.

5.3.3 Treatment of nonlinear and nonsmooth power functions

One of the difficulties in applying a linear MPC approach to the hydro power valley is the nonsmoothness of the power functions associated with the ducts C_1 and C_2. The nonsmoothness is caused by the fact that the flow through C_1 and C_2 can have two directions and the powers generated or consumed do not have equivalent coefficients. In order to handle this nonsmoothness, we use a double-flow technique, which means introducing two nonnegative positive variables to express the flow in $C_i, i = 1, 2$ at a sampling step k:

- $q_{C_{i\mathrm{P}}}(k)$: virtual flow such that C_i functions as a pump
- $q_{C_{i\mathrm{T}}}(k)$: virtual flow such that C_i functions as a turbine

When the solution is obtained, we combine the virtual flows to get the real flow through C_i:

$$q_{C_i}(k) = q_{C_{i\mathrm{T}}}(k) - q_{C_{i\mathrm{P}}}(k) \tag{5.14}$$

The introduction of virtual flows requires the input-matrices, B_i^{r}, to be augmented with two extra columns identical to the ones multiplying $q_{C_i}, i = 1, 2$ with the opposite sign to capture that pump action is also introduced with positive flow. The resulting reduced order model has 12 inputs instead of the original 10. Using the introduced flows $q_{C_{i\mathrm{P}}}$ and $q_{C_{i\mathrm{T}}}$, the power function (5.2) for subsystems 1 and 2 can be rewritten as

$$p_i(x(k), q(k)) = \left(k_{T_{C_i}} q_{C_{i\mathrm{T}}}(k) - k_{P_{C_i}} q_{C_{i\mathrm{P}}}(k)\right) \Delta x_{C_i}(k) + k_{T_i} q_{T_i}(k) \Delta x_{T_i}(k) \tag{5.15}$$

which is a continuous function.

The resulting nonlinear expression (5.15) can in turn be linearized around the steady-state solution $(x^{\mathrm{ss}}, q^{\mathrm{ss}})$. Since $q_{C_i}^{\mathrm{ss}} = 0$ for $i = 1, 2$ we get the following linear local power production/consumption approximation for subsystems $i = 1, 2$:

$$\begin{aligned}\hat{p}_i(x(k), q(k)) = \Delta x_{C_i}^{\mathrm{ss}} \begin{bmatrix} k_{T_{C_i}} & -k_{P_{C_i}} \end{bmatrix} \begin{bmatrix} q_{C_{i\mathrm{T}}}(k) \\ q_{C_{i\mathrm{P}}}(k) \end{bmatrix} + k_{T_i} q_{T_i}^{\mathrm{ss}} \left(\Delta x_{T_i}(k) - \Delta x_{T_i}^{\mathrm{ss}}\right) + \\ + k_{T_i} \Delta x_{T_i}^{\mathrm{ss}} \left(q_{T_i}(k) - q_{T_i}^{\mathrm{ss}}\right) + k_{T_i} q_{T_i}^{\mathrm{ss}} \Delta h_{T_i}^{\mathrm{ss}}\end{aligned}$$

For subsystems $i = 3, \ldots, 8$ we have the smooth power production expression (5.3) that can be directly linearized without introducing virtual flows:

$$\begin{aligned}\hat{p}_i(x(k), q(k)) = k_{D_i} q_{D_i}^{\mathrm{ss}} \Delta x_{D_i}^{\mathrm{ss}} + k_{D_i} q_{D_i}^{\mathrm{ss}} \left(\Delta x_{D_i}(k) - \Delta x_{D_i}^{\mathrm{ss}}\right) + \\ + k_{D_i} \Delta x_{D_i}^{\mathrm{ss}} \left(q_{D_i}(k) - q_{D_i}^{\mathrm{ss}}\right)\end{aligned}$$

5.4 HPV control using distributed MPC

The structured sparse model of the hydro power valley prompts us to consider designing a distributed MPC algorithm in which the subsystems' controllers cooperate locally to achieve a centrally optimal solution. In our distributed MPC framework, each subsystem only needs to communicate with a few neighboring subsystems, hence relaxing the communication requirements. Another advantage is that the computational tasks are divided into subtasks that are handled by local controllers in parallel, thus reducing the need for a powerful computing unit which is often needed in a centralized MPC implementation.

5.4.1 Distributed MPC algorithm using dual accelerated projected gradient method (DAPG)

In this section, we summarize the distributed dual accelerated projected gradient (DAPG) method that has been described in Chapter 3. The main idea is to exploit the problem structure of the dual problem such that the APG computations can be distributed to subsystems. Hence, the distributed algorithm effectively solves the centralized optimization problem.

The MPC optimization problem of the HPV will be cast into the following form

$$\begin{aligned} \min_{\mathbf{x},\mathbf{x}_\mathrm{a}} \ & \frac{1}{2}\mathbf{x}^T H\mathbf{x} + g^T\mathbf{x} + \gamma\|\mathbf{x}_\mathrm{a}\|_1 \\ \text{s.t. } & A_1\mathbf{x} = B_1 \\ & A_2\mathbf{x} \leq B_2 \\ & \mathbf{x}_\mathrm{a} = P\mathbf{x} - p \end{aligned} \tag{5.16}$$

where $\mathbf{x} \in \mathbb{R}^n$ includes all the states and inputs of the reduced model (5.12)–(5.13) and $\mathbf{x}_\mathrm{a} \in \mathbb{R}^m$ is the auxiliary decision vector that captures the errors of power-reference tracking. Denote M as the number of subsystems in the HPV, $\mathbf{x}$ is partitioned according to:

$$\mathbf{x} = [\mathbf{x}_1^T, \ldots, \mathbf{x}_M^T]^T, \tag{5.17}$$

and $\mathbf{x}_i \in \mathbb{R}^{n_i}$. Further, the matrix $H \in \mathbb{R}^{n\times n}$ is positive definite and block-diagonal with block-matrices $H_i \in \mathbb{R}^{n_i}$; the matrices $A_1 \in \mathbb{R}^{q\times n}$, $A_2 \in \mathbb{R}^{r\times n}$, and $P \in \mathbb{R}^{m\times n}$ have sparse structures.

As formulated in Chapter 3, the dual problem of (5.16) is to minimize the convex function:

$$\begin{aligned} f(z) &= J^\star(-\mathcal{A}^T z) + \mathcal{B}^T z \\ &= \frac{1}{2}(\mathcal{A}^T z + g)^T H^{-1}(\mathcal{A}^T z + g) + \mathcal{B}^T z \end{aligned} \tag{5.18}$$

with the following notation:

$$\mathcal{A} = [A_1^T \ A_2^T \ P^T]^T \qquad \mathcal{B} = [B_1^T \ B_2^T \ p^T]^T \qquad z = [\lambda^T \ \mu^T \ \nu^T]^T$$

where $\mathcal{A} \in \mathbb{R}^{(q+r+m)\times n}$, $\mathcal{B} \in \mathbb{R}^{q+r+m}$, and $z \in \mathbb{R}^{q+r+m}$. The set of feasible dual variables is defined as

$$Z = \mathbb{R}^q \times \mathbb{R}^r_{\geq 0} \times [-\gamma, \gamma]^m \tag{5.19}$$

We also name each line of $\mathcal{A}$ by using the following notations:

$$\mathcal{A} = [a_1 \ \cdots \ a_{q+r+m}]^T \tag{5.20}$$

with $a_l \in \mathbb{R}^n, l = 1, \ldots, q+r+m$. Furthermore, each $a_l, l = 1, \ldots, q+r+m$ is composed of the components that correspond to the variables of subsystems:

$$a_l = [a_{l1}^T, \ldots, a_{lM}^T]^T, \quad l = 1, \ldots, q+r+m \tag{5.21}$$

in which $a_{li} \in \mathbb{R}^{n_{\mathbf{x}_i}}, i = 1, \ldots, M$, where $n_{\mathbf{x}_i}$ is the size of the subsystem variable $\mathbf{x}_i$.

The implementation of the algorithm also requires defining the sets of neighbors $\mathcal{N}_i$ that have direct communications to a subsystem (including itself) and of the sets of constraints $\mathfrak{L}_i$ that are assigned to each subsystem $i \in \{1, \ldots, 8\}$. In general, the sparse structure of $\mathcal{A}$ determines how small $\mathcal{N}_i$ and $\mathfrak{L}_i$ are, and the smaller these sets are, the shorter range and less communication need to be used. For brevity and simplicity of the algorithm description, the notion $\mathfrak{L}_i$ in this chapter is the merger of $\mathfrak{L}_i^1$ and $\mathfrak{L}_i^2$ in Chapter 3, i.e., we will not distinguish between equality and inequality constraints here. A slightly simplified distributed dual APG algorithm for solving the dual problem is presented as follows.

Algorithm 5.1 ***Distributed Accelerated Proximal Gradient (DAPG)***

Initialize $z^0 = z^{-1}$ and $\mathbf{x}^{-1}$ with the last values from previous sampling time. For the first sampling time, these variables are initialized by zeros.

In every node, i, the following computations are performed in parallel and with synchronization:
For $k = 0, 1, 2, \ldots,$

1. *Compute*

$$\mathbf{x}_i^k = -H_i^{-1} \left(\sum_{j \in \mathcal{N}_i} \left(\sum_{l \in \mathfrak{L}_j} z_l^k a_{li} \right) + g_i \right) \tag{5.22}$$

$$\bar{\mathbf{x}}_i^k = \frac{2k+1}{k+2} \mathbf{x}_i^k - \frac{k-1}{k+2} \mathbf{x}_i^{k-1} \tag{5.23}$$

2. *Send $\bar{\mathbf{x}}_i^k$ to each $j \in \mathcal{N}_i$, receive $\bar{\mathbf{x}}_j^k$ from each $j \in \mathcal{N}_i$*

3. *Compute with each $l \in \mathfrak{L}_i$*

$$d_l^k = \sum_{j \in \mathcal{N}_i} a_{lj}^T \bar{\mathbf{x}}_j^k - b_l \tag{5.24}$$

$$z_l^{k+1} = z_l^k + \frac{k-1}{k+2}(z_l^k - z_l^{k-1}) + \frac{1}{L} d_l^k, \quad l \le q \tag{5.25}$$

$$z_l^{k+1} = \max\left\{0, z_l^k + \frac{k-1}{k+2}(z_l^k - z_l^{k-1}) + \frac{1}{L}d_l^k\right\}, \quad q < l \leq q + r \tag{5.26}$$

$$z_l^{k+1} = \min\left\{\gamma, \max\left[-\gamma, z_l^k + \frac{k-1}{k+2}(z_l^k - z_l^{k-1}) + \frac{1}{L}d_l^k\right]\right\}, \quad q + r < l \leq q + r + m \tag{5.27}$$

4. *Send* $\{z_l^{k+1}\}_{l\in\mathfrak{L}_i}$ *to each* $j \in \mathcal{N}_i$,
   receive $\{z_l^{k+1}\}_{l\in\mathfrak{L}_j}$ *from each* $j \in \mathcal{N}_i$.

5. *Each local controller checks the local termination criteria. If local termination criteria are satisfied, the algorithm stops, otherwise increase* k *and go to step 1) to start a new iteration.*

We note that the updates (5.25)–(5.27) involve L which is the smallest Lipschitz constant of ∇f and which is computed using the following formula (cf. Chapter 3, Proposition 3.1):

$$L = \|\mathcal{A}H^{-1}\mathcal{A}^T\|_2 \tag{5.28}$$

In other words, L is the largest singular value of the matrix $\mathcal{A}H^{-1}\mathcal{A}^T$, and $L > 0$ since H is positive definite and $\mathcal{A}$ is full-rank. We only need to compute L once and use it through the entire MPC control process.

5.4.2 HPV optimization problem formulation

In this section we will formulate an optimization problem of the form (5.16) that can be used for power-reference tracking in the HPV benchmark using MPC. We have obtained a linear discrete-time dynamical system (5.12)-(5.13) for the HPV with state variables x^r and control variables q^r. The constraints are upper and lower bounds on the outputs and inputs and can be found in Savorgnan and Diehl (2011). Using the transformation matrices (5.11) these constraints can readily be recast as linear constraints for the reduced order model variables x^r, q^r. The power-reference tracking problem formulation (5.1) specifies a quadratic cost on states and control variables and a 1-norm penalty on deviations from the provided power reference, p^{ref}. For a given control horizon N let the current MPC step $t = 0$ to simplify the expression, this optimization problem can be written as

$$\min_{\mathbf{x},\mathbf{x}_a} \sum_{t=0}^{N-1}\left\{\sum_{i=1}^{8}\left[x_i^r(k)^T Q_i x_i^r(k) + q_i^r(k)^T R_i q_i^r(k)\right] + \gamma\|x_a(k)\|_1\right\} \tag{5.29}$$

$$\begin{aligned}
\text{s.t.}\quad & (5.12), (5.13) && k = 0,\dots,N-1 && i = 1,\dots,8\\
& C_i^r x_i^r(k) \in \mathcal{Y}_i && k = 0,\dots,N-1 && i = 1,\dots,8\\
& q_i(k) \in \mathcal{Q}_i && k = 0,\dots,N-1 && i = 1,\dots,8\\
& x_a(k) = p^{\text{ref}}(k) - \textstyle\sum_{i=1}^{8}\hat{p}_i(x^r(k), q^r(k)) && k = 0,\dots,N-1
\end{aligned}$$

where $\mathcal{Y}_i$ and $\mathcal{Q}_i$ are sets representing the local output and input constraints, and the additional variable x_{a} captures the power-reference tracking mismatch.

There is an issue that prevents this optimization problem from being implemented in a distributed fashion using Algorithm 5.1, due to the fact that the original cost function is non-separable. Consequently, the power-reference constraints with the linearized power functions $\bar{p}_i$ include all variables. This implies that global communication is needed.

In order to obtain a suitable dual problem, we first need to reformulate the cost function (5.1) in a separable form. Next we present two ways of power reference division for the HPV problem. For the sake of brevity, we focus on one sampling step and drop the time index k. Thus for now our simplified objective is to decompose the following problem:

$$\min_{\{\mathbf{x}_i\}_{i=1,\dots,8}} \quad \left| p^{\mathrm{ref}} - \sum_{i=1}^{8} P_i \mathbf{x} \right| \tag{5.30}$$

with $\mathbf{x} = [\mathbf{x}_1^T, \dots, \mathbf{x}_8^T]^T$, and P_i the matrix coefficient such that the power function produced or consumed by each subsystem $\hat{p}_i(x^{\mathrm{r}}(k), q^{\mathrm{r}}(k))$ is linearized as $P_i\mathbf{x}(k)$.

Static power division

In order to avoid global communications, we can divide the total power reference into several sequences of local power references, i.e., the following cost function can be used instead of (5.30):

$$\min_{\{\mathbf{x}_i\}_{i=1,\dots,8}} \quad \sum_{j=1}^{\bar{M}} \left| p^{\mathrm{ref},j} - \sum_{i \in \mathcal{G}_j} P_i \mathbf{x} \right| \tag{5.31}$$

with $\bar{M}$ the number of separate power references such that $\sum_{j=1}^{\bar{M}} p^{\mathrm{ref},j} = p_{\mathrm{ref}}$, and $\mathcal{G}_j$ the group of subsystems that are assigned to track the power reference $p^{\mathrm{ref},j}$. The disadvantage of the power reference division is that the total power production may not be able to track the total power reference in some cases. This issue will be discussed with the illustrative results for the case study in Section 5.5.

We list here the strategies of dividing total power reference into different sequences that will be considered in the case study:

1. DIST–REF1: Use two sequences $p^{\mathrm{ref},1}$, $p^{\mathrm{ref},2}$ such that $p^{\mathrm{ref}} = p^{\mathrm{ref},1} + p^{\mathrm{ref},2}$. The total power production of subsystems 1, 3, 4, and 5 will track $p^{\mathrm{ref},1}$, while the sum of power produced by subsystems 2, 6, 7, and 8 will track $p^{\mathrm{ref},2}$.

2. DIST–REF2: Use 4 sequences: $p^{\mathrm{ref}} = p^{\mathrm{ref},1} + p^{\mathrm{ref},2} + p^{\mathrm{ref},3} + p^{\mathrm{ref},4}$, with $p^{\mathrm{ref},1}$ to be tracked by subsystems 1, 3, and 4, $p^{\mathrm{ref},2}$ to be tracked by subsystem 5, $p^{\mathrm{ref},3}$ to be tracked by subsystems 2, 6, and 7, and $p^{\mathrm{ref},4}$ to be tracked by subsystem 8.

3. LOCAL–REF: Use 8 sequences: $p^{\mathrm{ref}} = \sum_{i=1}^{8} p^{\mathrm{ref},i}$, with each local power reference $p^{\mathrm{ref},i}$ to be tracked by subsystem $i \in \{1, \dots, 8\}$.

In all these schemes, the local power reference sequences are computed by keeping the same proportion to the total power reference as in the steady-state operating conditions. The scheme LOCAL–REF means that each subsystem tracks a local power reference; however, this does not mean the setting is decentralized MPC, since the subsystems have to take into account the coupled dynamics. Note that the ways we define the tracking tasks with DIST–REF1 and DIST–REF2 are aimed at exploiting the existing structure of couplings in dynamics between subsystems.

Dynamic power division

The static power division essentially means that each subsystem always tracks a fraction of power reference that is equal to the proportion it produces in the steady-state condition. When the total power reference deviates significantly from the steady-state power, this idea may not work well since the proportional change of the local power reference can lead to sub-optimal performance. Inspired by an idea in Madjidian et al. (2011), we now introduce the dynamic power division, in which the subsystems have more flexibility in choosing the appropriate local power reference to be tracked. The main idea is that each subsystem will exchange an amount of power references with its direct neighbors.

Let us define for each pair (i, j) with $j \in \mathcal{N}_i$ a node that is in charge of determining the power exchange variable between subsystems i and j, denoted by δ_{ij} if node i is in charge and by δ_{ji} if node j is in charge [2]. Then for each subsystem we form the set [3]:

$$\Delta_i = \{j \mid j \in \mathcal{N}_i, i \text{ is in charge of } \delta_{ij}\}. \tag{5.32}$$

Now we replace (5.30) by the following cost function:

$$\min_{\{\mathbf{x}_i, \delta_i\}_{i=1,\dots,8}} \sum_{i=1}^{8} \left| p_i^{\text{ref}} + \sum_{j \in \Delta_i} \delta_{ij} - \sum_{j \in \mathcal{N}_i \setminus \Delta_i} \delta_{ji} - P_i \mathbf{x} \right| \tag{5.33}$$

with δ_i the vector containing all $\delta_{ij}, j \in \Delta_i$, and p_i^{ref} the nominal power reference for subsystem i. In words, the local power reference for each subsystem i deviates from the nominal value by adding the exchange amounts of the links that i manages and subtracting the exchange amounts of the links that affect i but are decided upon by its neighbors. Note that problem (5.33) has a sparse structure that complies with the existing sparse structure of the HPV system, i.e., this method does not expand the neighborhood set of each subsystem.

The advantage of this dynamic power division is that it makes use of the existing network topology to form a sparse cost function, and the total power reference is preserved even if the local power references can deviate from the nominal values, i.e., we always have:

$$\sum_{i=1}^{8} \left\{ p_i^{\text{ref}} + \sum_{j \in \Delta_i} \delta_{ij} - \sum_{j \in \mathcal{N}_i \setminus \Delta_i} \delta_{ji} \right\} = p^{\text{ref}} \tag{5.34}$$

[2] Note that here we discuss the power division for each sampling step, i.e., there are $\delta_{ij}(k)$ or $\delta_{ji}(k)$ with $k = 0, \dots, N-1$.

[3] A simple way is to let the subsystem with smaller index lead the exchange, i.e., $\Delta_i = \{j | j \in \mathcal{N}_i, j > i\}$.

Now that we have a separable cost function by using either a static or a dynamic power division technique, we can form a separable dual problem and apply a distributed optimization algorithm for the dual problem. In the case study of Section 5.5, we will choose the distributed *dual accelerated proximal gradient* (DAPG) algorithm introduced in Chapter 3, as it was designed to treat the problems with mixed absolute and quadratic terms like the HPV problem in a more efficient way than translating the problem into a QP, and this method also has fast convergence rate.

5.5 Comparison of MPC schemes

We performed numerical simulations of the HPV using centralized MPC, decentralized MPC, and distributed MPC based on the DAPG algorithm with the 3 static power division schemes: DIST–REF1, DIST–REF2, and LOCAL–REF, and the dynamic power division scheme referred to as DYN–REF. For centralized and decentralized MPC, the optimization problems are transformed into QPs and are solved by the solver *quadprog* in the Optimization Toolbox of MATLAB (The Mathworks, 2012). For distributed MPC, at each MPC step, our own DAPG solver, also implemented in MATLAB, iterates until a small tolerance is achieved; thus, we almost get the exact solution for the optimization problem. The closed-loop simulation is obtained by simulating the computed inputs with the original nonlinear continuous-time model, using the function *ode15* of MATLAB. The simulations were implemented on a PC running MATLAB on Linux with an Intel(R) Core(TM)2 Duo CPU running at 2.33 GHz and with 2 GB RAM.

5.5.1 Performance comparison

The power reference tracking results are plotted in Figures 5.2(f)–5.2(a). We can see the trade-off due to the division in power reference: centralized MPC achieves the best tracking performance at the cost of using global communications, while distributed MPC with static power division schemes DIST1–REF, DIST2–REF, and LOCAL–REF shows deterioration of the tracking performance. Another interesting point is that the scheme DIST–REF2 achieves a better tracking performance than LOCAL–REF, while they use the same communication structure. This observation suggests to consider finding the division of power reference that resembles the structure of dynamical couplings between subsystems, i.e., defining groups for tracking group power references such that any pair of subsystems in a group are neighbors of each other. The tracking performance of decentralized MPC is very poor, due to the lack of communications. The best result, in view of both achieving performance and using local communications only, is distributed MPC with the dynamic power division scheme DYN–REF, which achieves almost the same tracking performance as centralized MPC, while relying on the same communication structure as the LOCAL–REF scheme (see also the comparison of communication requirements).

Figures 5.3 and 5.4 show the evolution of the inputs and outputs and the corresponding constraints with the scheme DYN–REF as an example. These results for the cases of DIST–REF1, DIST–REF2, and LOCAL–REF are slightly different, however all the constraints on the inputs and outputs are also satisfied. We note that this is guaranteed due to the fact that constraints are not relaxed, and the couplings in dynamics are taken into account.

5.5.2 Computational efficiency

In Figure 5.5, we plot the comparison of computation time of the solvers *quadprog* and our own solver *DAPG* (recall that both are implemented in MATLAB), for solving the same optimization problem for centralized MPC at each sampling step. Figure 5.5 shows that the total computation time of the *DAPG* solver is much lower than the computation time of *quadprog*, in average the *DAPG* solver reduces 80% computation time of *quadprog*. This difference reflects the fast convergence rate of the DAPG algorithm and the efficiency of dealing with the absolute term in the cost function.

5.5.3 Communication requirements

In Table 5.1, we provide the comparison of the *communication neighborhood* sets between the distributed MPC schemes. The communication neighborhood set of a subsystem i, denoted by $\mathcal{M}_i$, indicates which subsystems that subsystem i needs to communicate with. Note that these communication neighborhood sets do not only contain the neighbors in the sets $\mathcal{N}_i$ coming from the coupling dynamics, but can be extended due to the additional coupled constraints that are introduced by the power reference division schemes. In the scheme DIST–REF1, the sets $\mathcal{M}_i$ are larger than $\mathcal{N}_i$, since we form power tracking groups that involve subsystems which are not dynamically coupled, e.g., between subsystems 1 and 5, 2 and 8. With the schemes DIST–REF2, LOCAL–REF, and DYN–REF, the sets $\mathcal{M}_i$ are the same as $\mathcal{N}_i$.

Combining the comparison of communication requirement and performance, distributed MPC with the dynamic power division scheme DYN–REF would be the best choice for the total power reference tracking problem of the HPV, as it achieves almost the same optimal solution as centralized MPC, while allowing a distributed implementation using the existing dynamical coupling structure. Among distributed MPC with static power division schemes, the scheme LOCAL–REF is the most naive idea and its performance suffers the most; the scheme DIST–REF1 shows a fairly good tracking performance of the total power reference; however, it employs a more complex communication structure; the scheme DIST–REF2 provides a good balance between tracking performance and limiting communication requirement, and it achieves a much better result than the scheme LOCAL–REF while sharing the same communication structure with this basic scheme. It should also be noted that decentralized MPC is not recommended unless communication is prohibited, otherwise the performance will deteriorate significantly.

5.6 Conclusions and recommendations for future research

The distributed MPC scheme using distributed accelerated proximal gradient algorithm introduced in Chapter 3 has been applied to the power reference tracking problem of the HD-MPC hydro power valley benchmark. Several distributed schemes have been compared to centralized and decentralized MPC methods. We have provided relaxations and approximations for the original nonlinear and non-smooth problem as well as modifications of the centralized power reference that lead to separable cost functions. The simulation results show that the introduced approximate problem formulation captures the behaviour of

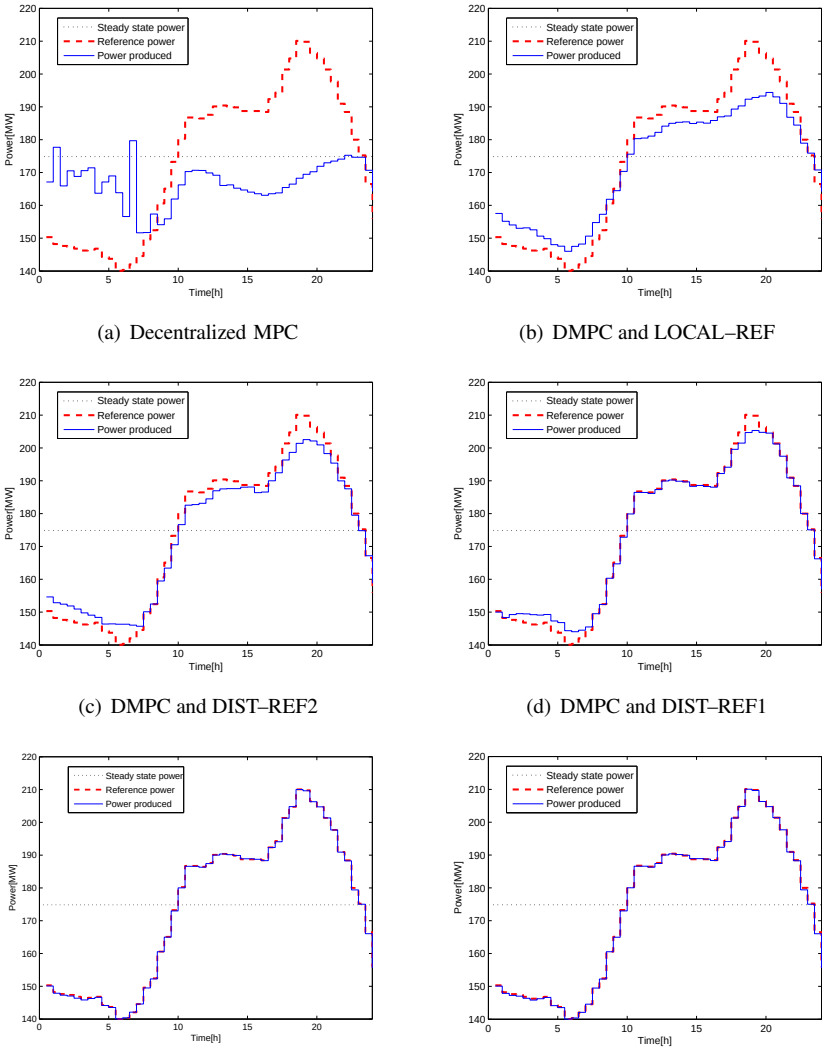
Table 5.1: *Communication neighborhoods of subsystems ($\mathcal{M}_i$)*

i	DMPC **DIST–REF1**	DMPC **DIST–REF2**	DMPC **LOCAL–REF**	DMPC **DYN–REF**
1	$\{1,3,4,5\}$	$\{1,3,4\}$	$\{1,3,4\}$	$\{1,3,4\}$
2	$\{2,6,7,8\}$	$\{2,6,7\}$	$\{2,6,7\}$	$\{2,6,7\}$
3	$\{3,1,4,5\}$	$\{3,1,4\}$	$\{3,1,4\}$	$\{3,1,4\}$
4	$\{4,1,3,5\}$	$\{4,1,3,5\}$	$\{4,1,3,5\}$	$\{4,1,3,5\}$
5	$\{5,1,3,4,6\}$	$\{5,4,6\}$	$\{5,4,6\}$	$\{5,4,6\}$
6	$\{6,2,7,8,5\}$	$\{6,2,7,5\}$	$\{6,2,7,5\}$	$\{6,2,7,5\}$
7	$\{7,2,6,8\}$	$\{7,2,6,8\}$	$\{7,2,6,8\}$	$\{7,2,6,8\}$
8	$\{8,2,6,7\}$	$\{8,7\}$	$\{8,7\}$	$\{8,7\}$

Centralized MPC uses global communications: $\mathcal{M}_i = \{1,\ldots,8\}, \forall i$
Decentralized MPC does not use communications: $\mathcal{M}_i = \{i\}, \forall i$

the system well and that very good control performance is achieved with distributed MPC with the dynamic power division technique. Further, a comparison of computational time shows that the proposed DAPG algorithm is more efficient than the quadratic program solver *quadprog* for centralized MPC.

As the next step before implementation in real plants, the proposed distributed MPC approach should be tested against different hydraulic scenarios. To cope with varying waterflows entering the system, these should be estimated and compensated for. Further a weather model could be included that estimates the future in-flows to the HPV system.



(a) Decentralized MPC

(b) DMPC and LOCAL–REF

(c) DMPC and DIST–REF2

(d) DMPC and DIST–REF1

(e) DMPC and DYN–REF

(f) Centralized MPC

Figure 5.2: *Comparison of power reference tracking performance using centralized MPC, DMPC, and decentralized MPC.*

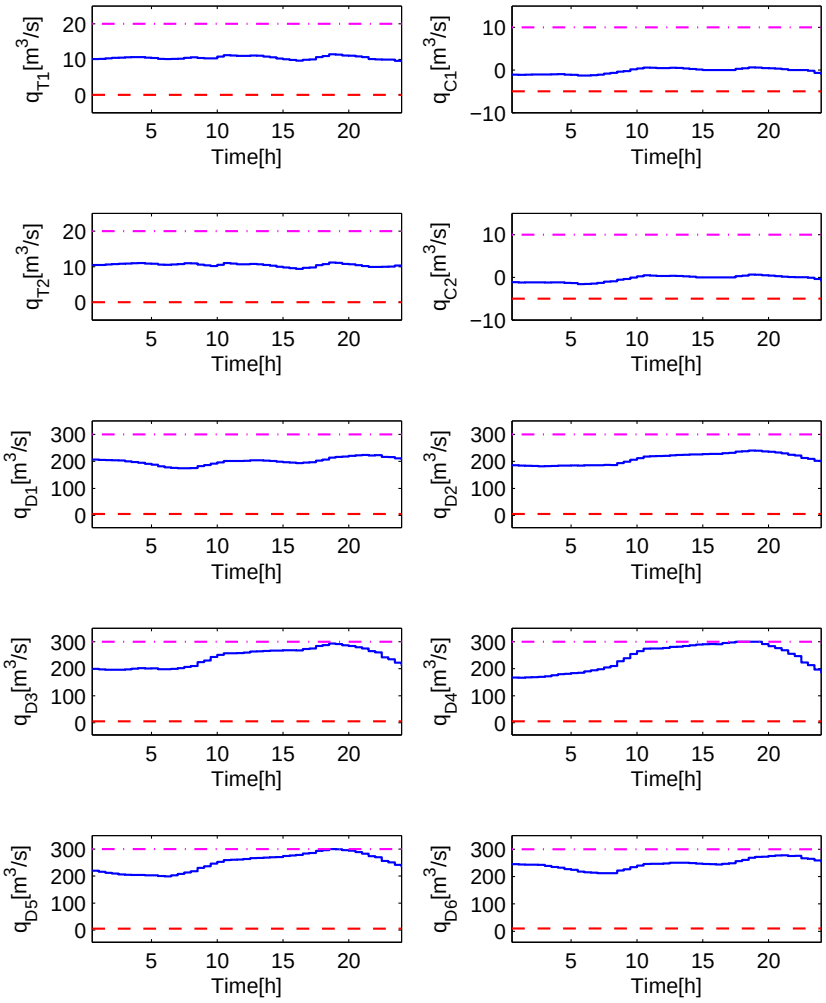


Figure 5.3: *Input constraint satisfaction with DMPC based on DAPG algorithm with the dynamic power division scheme DYN–REF. Dash-dotted lines: upper bounds, dashed lines: lower bounds.*

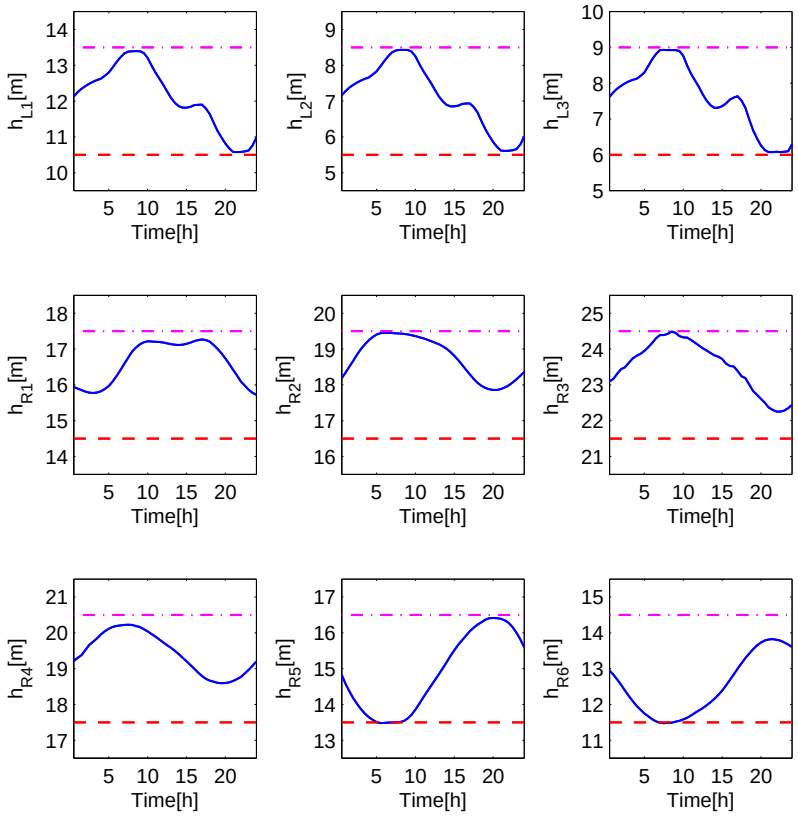


Figure 5.4: *Output constraint satisfaction with DMPC based on DAPG algorithm with the dynamic power division scheme DYN–REF. Dash-dotted lines: upper bounds, dashed lines: lower bounds.*

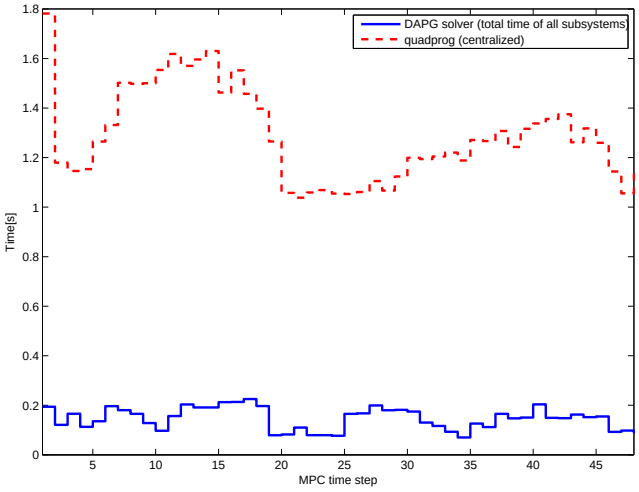


Figure 5.5: *Comparison of computation time to solve the MPC optimization problem per each sampling step*

Chapter 6

Conclusions and recommendation for future research

We provide a summary of the methods presented in this thesis and highlight the main contributions. We conclude the thesis by giving recommendations for extending our research and for exploring related topics in distributed MPC.

6.1 Summary and main contributions of the thesis

6.1.1 Summary of the proposed methods

In this section, we summarize the properties of the algorithms that have been presented in this thesis, and discuss the advantages as well as disadvantages of these methods.

For each method, we state the problem classes that the method is applicable to, and then we list the properties of the optimization algorithm and the closed-loop MPC process based on that method.

In Chapter 2, a distributed decomposition scheme based on Han's parallel method is presented. This scheme provides a distributed approach to solve the centralized optimization problem corresponding to an MPC formulation for linear systems. The algorithm can be used when the MPC optimization problem is a strongly convex quadratic problem, i.e., the cost function is positive definite, and subject to linear constraints. Another condition is that the constraint coupling is sparse, i.e., each subsystem only couples with a small number of other subsystems through constraints. We have shown that the distributed Han's algorithm generates the same solution as the centralized Han's algorithm, hence its convergence is inherited from the centralized counterpart. The distributed MPC scheme using distributed Han's algorithm guarantees feasibility and stability upon optimality of the solution at every sampling step.

In Chapter 3, a distributed accelerated proximal gradient (DAPG) method is developed that also provides a distributed approach for solving the centralized MPC optimization problem for linear systems. This scheme can handle a larger problem class than distributed

Han's method, since there can be an 1-norm term in the cost function in addition to the strongly convex quadratic one in the cost function. The constraints are required to be linear, and also to introduce sparse couplings between subsystems. We have shown that the dual problem has a Lipschitz continuous gradient and the maximum Lipschitz constant of the gradient is used to derive an optimal step size of the algorithm. One merit of the DAPG algorithm is that it exhibits fast convergence, i.e., the solution error is reduced with the rate of $O(\frac{1}{k^2})$ where k is the iteration index. This convergence rate is the best convergence rate for a gradient method in the literature, and in comparison, the distributed Han's method is similar to the classical proximal gradient algorithm which has a convergence rate of $O(\frac{1}{k})$ only. For the problem class considered, the DAPG algorithm can obtain a centralized solution by using a distributed implementation. The feasibility and stability of the closed-loop MPC are also established upon optimality.

In Chapter 4, a constraint tightening approach is employed that leads to two hierarchical MPC methods that provide feasible solutions after a finite number of iterations, even in the framework of dual decomposition. The two hierarchical methods are applicable to optimization problems with convex decoupled cost functions and weakly coupled, linear inequality constraints. Note that all equality constraints should be eliminated in advance, so that the constraint set of the MPC optimization problem has non-empty interior. At the first MPC step, it is necessary to provide a Slater vector, i.e., a point in the interior of the feasible set of the optimization problem; however for the subsequent steps the algorithm can generate suitable Slater vectors itself. The hierarchical algorithms can generate feasible primal solutions after a finite number of iterations using a dual conjugate gradient method (HPF-DCG) and using a dual approximate gradient with Jacobi methods (HPF-DAG). Hence they avoid solving the dual problem until an optimal solution is found. The two algorithms use different sub-algorithms to get the primal update at each iteration: the algorithm HPF-DCG uses a conjugate gradient method to obtain an exact solution of the primal problem, while the algorithm HPF-DAG uses the Jacobi iteration and obtains an inexact solution instead. Both algorithms achieve closed-loop MPC stability by guaranteeing a monotonical decrease of the cost function, which then can act as a Lyapunov function.

In Chapter 5, we use the distributed approach based on accelerated proximal gradient method (DAPG) for designing a distributed MPC controller for a Hydro Power Valley benchmark, and we compare the results against centralized MPC and decentralized MPC. We use decentralized model order reduction to obtain a suitable model for applying these three control methods. Furthermore, we propose a dynamic division scheme to decouple the cost function, thus obtaining an optimization problem that is suitable for distributed optimization. The results show that the distributed MPC algorithm can handle the economic MPC problem of the Hydro Power Valley benchmark efficiently. Distributed MPC provides a nearly optimal performance that resembles the centralized MPC performance, and outperforms decentralized MPC. Moreover, distributed MPC using DAPG method is more computationally efficient than centralized MPC using a standard quadratic programming solver.

6.1.2 Main contributions

In this section, we highlight the main contributions of the work presented in this thesis:

1. We have developed a distributed optimization method based on Han's parallel algorithm; the distributed version is able to generate the same iterate as the parallel (centralized) counterpart, and is guaranteed to converge to the centralized solution.

2. We have developed a distributed version of the accelerated proximal gradient method that can handle optimization problems with mixed 1-norm and 2-norm penalty terms in the cost function and with linear constraints having a sparse coupling structure. The distributed accelerated proximal gradient algorithm achieves the best convergence rate of a gradient method.

3. We have proposed an approach using constraint tightening that leads to two hierarchical algorithms for solving the MPC optimization problem based on dual decomposition. The algorithms can be terminated after a finite number of iterations, but still provide feasible solutions to the MPC problem. The algorithms also guarantee a monotonic decrease of the cost function, thus leading to MPC stability.

4. We have performed a numerical study of hydro power valley benchmark, where the nonlinear model is treated with linearization and model order reduction that lead to a simplified linear model to which a distributed MPC controller is applied. The numerical simulations show that distributed MPC can obtain comparable performance with centralized MPC, while the total computational time of distributed MPC can be significantly lower than a standard algorithm for centralized MPC.

6.2 Recommendations for future research

6.2.1 Further improvements to distributed MPC design

We have proposed methods to deal with the MPC optimization problem in a distributed way. However, the properties of the MPC controllers depend on several assumptions of which the realization is beyond the scope of this work. In order to enable a procedure for distributed MPC design and implementation, the following open issues should be addressed next:

- **Computation of structured terminal penalty matrix**: Stability must be ensured for the distributed MPC algorithm when using a dual decomposition approach. The standard MPC problem formulations that guarantee stability apply a terminal point constraint which uses the cost function as a Lyapunov candidate function, and the dual-mode MPC formulation which uses the terminal cost as a Lyapunov candidate function. The former approach is quite conservative, though it results in a sparse optimization problem that is favorable for distributed MPC. The latter approach is less conservative, however it requires computing a penalty matrix as a solution of the Riccati equation. Moreover, the penalty matrix should have a sparse structure in order to facilitate distributed MPC. As of present, a distributed algorithm for computing a structured solution of the Riccati equation is still missing. We intend to generalize the parallel algorithm to solve the Riccati equation of (Gajic and Shen, 1993, Chapter 7), and then approximate the solution with a diagonal banded matrix so that the cost function has some kind of sparsity structure.

- **Design of decentralized stabilizing controllers and positively invariant sets**: The formulation of dual-mode MPC also requires using a terminal stabilizing controller when the terminal state is driven into an invariant set of that controller. In the context of distributed MPC, we need to design decentralized stabilizing controllers and the corresponding local invariant sets. Although there are results on these topics (Kerrigan, 2000, Blanchini and Miani, 2008), these methods require centralized computation, there is still neither a distributed scheme available to construct the decentralized terminal controllers and local invariant sets, nor a verification method to check whether these results actually exist. We suggest to tackle this problem by a partitioning approach that neglects the coupling between subsystems, thus allowing terminal stabilizing controllers and positively invariant sets to be designed by each local controller, and then verify a posteriori the applicability of the results when the dynamical coupling is taken into account.

6.2.2 Towards real-life implementations of distributed MPC

The ultimate research goal is to apply distributed MPC in real-world applications, where the control objective is suitable for MPC, and the problem is too complex to be handled in a centralized manner. To the author's best knowledge, there has not been a complex system that is operated using distributed MPC in practice yet, since there are several topics that need further research:

- **Non-cooperative approach for large-scale infrastructure systems**: A potential application field for distributed MPC is large-scale infrastructure systems, which can be wide-area networks, e.g., power transmission networks, open water networks, or traffic networks, where different agents participate in a non-cooperative manner. In such scenario, the cooperative approach taken in this thesis may not be applicable, and hence a game theory-based approach is worth considering. Relevant topics for further research are: how to set the regulation laws in a network such that the agents will contribute to the global performance while they are optimizing their own benefits, how to quantify the sub-optimality of the non-cooperative distributed solutions, and which level of cooperation should be imposed to balance the global and individual benefits.

- **Reconfigurability, robustness of networks controlled by distributed MPC**: Another application field with much prospect is the class of large-scale systems that operate in a fixed area, belonging to the same owner, e.g., transportation system in a large storage area or a factory, resource handling in a large business system. Since in this area all subsystems can be designed by a single designer, the cooperative frameworks proposed in this thesis are applicable. However further steps are needed before practical implementation. The open questions include: how a subsystem is plugged in or detached out of the whole network, how to make use of the knowledge about repetitive patterns like daily task loads or periodic disturbances, how to ensure robustness of the whole or part of the network against disturbances such as model mismatch, communication link missing, or subsystem failure.

6.2.3 Related topics in distributed MPC

The work presented in this thesis can also be extended to related topics that were not carried out in the time frame of research. These ideas can be developed as open directions for further research in distributed MPC.

- **Distributed MPC using primal decomposition-based methods**: The distributed and hierarchical MPC approaches proposed in this thesis are based on dual decomposition, i.e., aim to exploit the separable property of the dual problem. For some classes of problems, even the primal problem can have structures that foster decomposition; hence primal decomposition approaches can be used then. The research on primal decomposition for large-scale MPC optimization problems can be built upon the well-known Gauss-Seidel (serial) or Jacobi (parallel) distributed algorithms.
- **Formulation of closed-loop distributed MPC using a robustness approach**: In most cases distributed MPC only achieves sub-optimal results. Hence, there is the need to quantify the sub-optimality of a distributed MPC scheme with respect to the number of iterations that subsystems exchange information, and find some bounds for the sub-optimality. Those bounds, being available in advance, can be used in the design phase with a robustness approach, so that MPC stability is guaranteed.
- **Dealing with periodic behaviors**: In many applications the systems operate under periodically changing inputs or conditions, e.g., the demands for resources such as electricity or irrigation water in infrastructure networks. Hence, the design of distributed MPC may not need to start off solving an optimization problem, but instead efficient controllers can be constructed using self-adapting methods of each subsystem based on the nominal globally-designed profile. By judging the difference between the current situation and the nominal case, local controllers can adapt the stored MPC solutions to get a feasible solution that can be implemented locally.
- **Distributed moving-horizon state estimation (MHSE)**: The problem of distributed MHSE shares a significant similarity with distributed MPC, where the main task is to solve an optimization problem online with respect to the estimated/predicted variables in a finite horizon, and the major challenge is to generate converging/stabilizing solutions in a distributed manner. Hence, distributed optimization techniques can also be applied to distributed MHSE in a similar way like in distributed MPC, with appropriate re-definition of local estimators and local variables.
- **Partitioning of large-scale interconnected systems**: In some large-scale systems such as a power transmission network of a region or a water distribution network of a city, the whole system could be too complicated to build a centralized dynamical model, instead the component models are constructed and then connected to each other. Hence, the definition of subsystems in these large-scale systems is not straightforward. One needs to find an appropriate method to partition the system, resulting in subsystems that have their own inputs and outputs, and there are weak couplings between subsystems. Graph theory could be used to obtain such partition.

Symbols and Abbreviations

Constants and Indices

i, j	Indices indicating different subsystems
k or t	Discrete time index
p	Iteration index in optimization algorithms
N	Prediction horizon of MPC
T_s	Sampling time of discrete-time systems
I_n	Identity matrix of dimension $n \times n$
$\mathbf{1}_n$	Column vector of dimension n with all entries equal to 1

Network notations

M	Number of nodes (usually subsystems)
$\mathcal{N}^i$	Neighborhood set of subsystem i, containing indices of itself and subsystems that have direct couplings with it
$\mathfrak{I}^i$	Index matrix of subsystem i (a selection matrix to keep the variables of subsystem i in the aggregate variable, and to replace the other entries in the aggregate variable by zeros)

Dynamical Systems

x	State vector
u	Input vector
$\mathbf{x}$	Prediction vector of states, or all the variables in the optimization problem
$\mathbf{u}$	Prediction vector of inputs
$\mathcal{X}$	Constraint set for the states
$\mathcal{U}$	Constraint set for the control inputs
X_f	Constraint set for the terminal states, i.e., states at the last predicted step
A, B	Matrices of state-space dynamics

Notations in optimization problems

H	Hessian matrix of a quadratic cost function

f	Cost function of an optimization problem
∇f	Gradient of function f
f^*	Conjugate function of function f
g, h	Constraint functions of an optimization problem
q	Dual function of an optimization problem
$\mathbf{\Omega}$	Domain for an optimization problem (Cartesian set)
$\mathcal{L}$	Lagrangian of an optimization problem
$\mathfrak{L}_i$	Set of constraint indices that are assigned to subsystem i for local computations
$\mathfrak{L}_{\mathcal{N}^i}$	Union of all sets $\mathfrak{L}_j$ with $j \in \mathcal{N}^i$

Abbreviations

MPC	Model Predictive Control
DMPC	Distributed Model Predictive Control
QP	Quadratic Program

Bibliography

A. Alessio and A. Bemporad. Decentralized model predictive control of constrained linear systems. In *European Control Conference*, pages 2813–2818, 2007.

A. Alessio and A. Bemporad. Stability conditions for decentralized model predictive control under packet drop communication. In *Proceedings of American Control Conference*, pages 3577–3582, Seattle, WA, June 2008.

A. Anand, G. Joshua, S. Sundaramoorthy, and L. Samavedham. Coordinating multiple model predictive controllers for multi-reservoir management. In *Proceedings of the 2011 IEEE International Conference on Networking, Sensing and Control*, pages 1–6, Delft, The Netherlands, Apr. 2011.

M. Arnold, R. R. Negenborn, G. Andersson, and B. De Schutter. Distributed predictive control for energy hub coordination in coupled electricity and gas networks. In R. R. Negenborn, Z. Lukszo, and H. Hellendoorn, editors, *Intelligent Infrastructures*, pages 235–273. Springer, 2010.

K. Arrow, L. Hurwicz, and H. Uzawa. *Studies in Linear and Nonlinear Programming*. Stanford University Press, Stanford, CA, 1958.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, Oct 2009.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Upper Saddle River, NJ, 1989.

F. Blanchini and S. Miani. *Set-Theoretic Methods in Control*. Birkhauser, Boston, 2008.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, MA, 2004.

E. Camponogara and S. Talukdar. Distributed model predictive control: Synchronous and asynchronous computation. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 37(5):732–745, Sept. 2007.

E. Camponogara, D. Jia, B. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 22(1):44–52, Feb. 2002.

P. Carpentier and G. Cohen. Applied mathematics in water supply network management. *Automatica*, 29(5): 1215 – 1250, Sept. 1993.

G. Danzig and P. Wolfe. The decomposition algorithm for linear programming. *Econometrica*, 29(4):767–778, Oct. 1961.

D. Doan, T. Keviczky, I. Necoara, M. Diehl, and B. De Schutter. A distributed version of Han's method for DMPC using local communications only. *Control Engineering and Applied Informatics*, 11(3):6–15, Sept. 2009.

M. Doan, T. Keviczky, and B. De Schutter. A distributed optimization-based approach for hierarchical MPC of large-scale systems with coupled dynamics and constraints. In *Proceedings of the 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 5236–5241, Orlando, Florida, Dec. 2011a.

M. D. Doan, T. Keviczky, and B. De Schutter. An improved distributed version of Han's method for distributed MPC of canal systems. In *Proceedings of the 12th symposium on Large Scale Systems: Theory and Applications*, Villeneuve d'Ascq, France, July 2010.

M. D. Doan, T. Keviczky, and B. De Schutter. A dual decomposition-based optimization method with guaranteed primal feasibility for hierarchical MPC problems. In *Proceedings of the 18th IFAC World Congress*, pages 392–397, Milan, Italy, Aug.–Sept. 2011b.

M. D. Doan, T. Keviczky, and B. De Schutter. An iterative scheme for distributed model predictive control using Fenchel's duality. *Journal of Process Control*, 21(5):746–755, 2011c. Special Issue on Hierarchical and Distributed Model Predictive Control.

M. D. Doan, P. Giselsson, T. Keviczky, B. De Schutter, and A. Rantzer. A distributed proximal gradient algorithm for DMPC of a Hydro Power Valley. Technical report, 2012. Submitted to Control Engineering Practice.

W. B. Dunbar and R. M. Murray. Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica*, 42(4):549–558, Apr. 2006.

H. Fawal, D. Georges, and G. Bornard. Optimal control of complex irrigation systems via decomposition-coordination and the use of augmented Lagrangian. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 3874–3879, San Diego, CA, Oct. 1998.

W. Findeisen. *Control and Coordination in Hierarchical Systems*. International Series on Applied Systems Analysis. Wiley, 1980.

Z. Gajic and X. Shen. *Parallel Algorithms for Optimal Control of Large Scale Linear Systems*. Springer New York, Inc., Secaucus, NJ, 1993.

J. Gauvin. A necessary and sufficient regularity condition to have bounded multipliers in nonconvex programming. *Mathematical Programming*, 12(1):136–138, 1977.

D. Georges. Decentralized adaptive control for a water distribution system. In *Proceedings of the Third IEEE Conference on Control Applications*, pages 1411–1416 vol.2, Glasgow, UK, Aug. 1994.

P. Giselsson and A. Rantzer. Distributed model predictive control with suboptimality and stability guarantees. In *Proceedings of the 49th Conference on Decision and Control*, pages 7272–7277, Atlanta, GA, Dec. 2010.

P. Giselsson, M. D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer. Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*. Accepted for Publication.

M. Gomez, J. Rodellar, and J. A. Mantecon. Predictive control method for decentralized operation of irrigation canals. *Applied Mathematical Modelling*, 26(11):1039–1056, 2002.

S. Gugercin and A. C. Antoulas. A survey of model reduction by balanced truncation and some new results. *International Journal of Control*, 77(8):748–766, 2004.

S.-P. Han and G. Lou. A parallel algorithm for a class of convex programs. *SIAM Journal on Control and Optimization*, 26(2):345–355, Mar. 1988.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, Feb. 1990.

J. Igreja and J. Lemos. Nonlinear model predictive control of a water distribution canal pool. In L. Magni, D. Raimondo, and F. Allgower, editors, *Nonlinear Model Predictive Control*, volume 384 of *Lecture Notes in Control and Information Sciences*, pages 521–529. Springer, Berlin / Heidelberg, Germany, 2009.

J. Igreja, F. Cadete, and J. Lemos. Application of distributed model predictive control to a water delivery canal. In *Proceedings of the 19th Mediterranean Conference on Control & Automation*, pages 682–687, Corfu, Greece, June 2011.

D. Jia and B. Krogh. Distributed model predictive control. In *Proceedings of American Control Conference*, pages 2767–2772 vol.4, 2001.

D. Jia and B. Krogh. Min-max feedback model predictive control for distributed control with communication. In *Proceedings of American Control Conference*, pages 4507–4512 vol.6, Pittsburgh, PA, 2002.

S. S. Keerthi and E. G. Gilbert. Optimal infinite-horizon feedback control laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations. *Journal of Optimization Theory and Applications*, 57(2):265–293, May 1988.

E. C. Kerrigan. *Robust Constraint Satisfaction: Invariant Sets and Predictive Control*. PhD thesis, Department of Engineering, University of Cambridge, UK, November 2000.

T. Keviczky, F. Borrelli, and G. J. Balas. Decentralized receding horizon control for large scale dynamically decoupled systems. *Automatica*, 42(12):2105–2115, Dec. 2006.

M. Kögel and R. Findeisen. A fast gradient method for embedded linear predictive control. In *Proceedings of the 18th IFAC World Congress*, pages 1362–1367, Milan, Italy, 2011.

Y. Kuwata, A. Richards, T. Schouwenaars, and J. P. How. Distributed robust receding horizon control for multi-vehicle guidance. *IEEE Transactions on Control Systems Technology*, 15(4):627–641, 2007.

S. Li, Y. Zhang, and Q. Zhu. Nash-optimization enhanced distributed model predictive control applied to the shell benchmark problem. *Information Sciences*, 170(2-4):329–349, Feb. 2005.

X. Litrico and V. Fromion. *Modeling and Control of Hydrosystems*. Springer, London, UK, 2009.

J. M. Maciejowski. *Predictive Control with Constraints*. Prentice-Hall, Harlow, UK, 2002.

D. Madjidian, K. Martensson, and A. Rantzer. A distributed power coordination scheme for fatigue load reduction in wind farms. In *Proceedings of American Control Conference*, pages 5219–5224, San Francisco, CA, 2011.

I. Mareels, E. Weyer, S. K. Ooi, M. Cantoni, Y. Li, and G. Nair. Systems engineering for irrigation systems: Successes and challenges. *Annual Reviews in Control*, 29(2):191–204, 2005.

D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(7):789–814, June 2000.

M. Mercangoz and F. J. Doyle III. Distributed model predictive control of an experimental four-tank system. *Journal of Process Control*, 17(3):297–308, Mar. 2007.

M. Mesarovic, D. Macko, and Y. Takahara. *Theory of Hierarchical Multilevel Systems*. Academic Press, New York, NY, 1970.

B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, 26(1):17–32, 1981.

I. Necoara and J. A. K. Suykens. Application of a smoothing technique to decomposition in convex optimization. *IEEE Transactions on Automatic Control*, 53(11):2674–2679, Dec. 2008.

I. Necoara, D. Doan, and J. Suykens. Application of the proximal center decomposition method to distributed model predictive control. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 2900–2905, Cancun, Mexico, Dec. 2008.

E. Nederkoorn, J. Schuurmans, and W. Schuurmans. Continuous nonlinear model predictive control of a hybrid water system - Application of DNPC to a Dutch polder. In *Proceedings of the 2011 IEEE International Conference on Networking, Sensing and Control*, pages 209–214, Delft, The Netherlands, Apr. 2011.

A. Nedic and A. Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization*, 19(4):1757–1780, Nov. 2009.

R. Negenborn, B. De Schutter, and J. Hellendoorn. Multi-agent model predictive control for transportation networks: Serial versus parallel schemes. *Engineering Applications of Artificial Intelligence*, 21(3):353–366, Apr. 2008.

R. R. Negenborn, P. J. van Overloop, T. Keviczky, and B. De Schutter. Distributed model predictive control for irrigation canals. *Networks and Heterogeneous Media*, 4(2):359–380, June 2009.

A. S. Nemirovskii and D. B. Yudin. *Informational Complexity and Efficient Methods for Solution of Convex Extremal Problems*. Wiley, New York, NY, 1983.

Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Y. Nesterov. On an approach to the construction of optimal methods of minimization of smooth convex functions. *Ekonomika i Matematicheskie Metody*, 24:509–517, 1988. In Russian.

Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course (Applied Optimization)*. Springer Netherlands, Dordrecht, Netherlands, 1st edition, 2004.

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

F. Petrone. Model predictive control of a hydro power valley. Master's thesis, Politecnico di Milano, Italy, 2010.

PEW Center on Global Climate Change. Hydropower - Climate TechBook, Sept. 2011. Available: http://www.c2es.org/docUploads/Hydropower_0.pdf.

S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, July 2003.

J. B. Rawlings and D. Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, Madison, WI, 2009.

J. B. Rawlings and B. T. Stewart. Coordinating multiple optimization-based controllers: New opportunities and challenges. *Journal of Process Control*, 18(9):839–845, Oct. 2008.

A. Richards and J. How. Robust distributed model predictive control. *International Journal of Control*, 80(9): 1517–1531, Sept. 2007.

S. Richter, C. Jones, and M. Morari. Real-time input-constrained MPC using fast gradient methods. In *Proceedings of the 48th Conference on Decision and Control*, pages 7387–7393, Shanghai, China, Dec. 2009.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.

A. Sahin and M. Morari. Decentralized model predictive control for a cascade of river power plants. In R. R. Negenborn, Z. Lukszo, and H. Hellendoorn, editors, *Intelligent Infrastructures*, volume 42 of *Intelligent Systems, Control and Automation: Science and Engineering*, pages 463–485. Springer Netherlands, 2010.

C. Savorgnan and M. Diehl. Control benchmark of a hydro-power plant. Technical report, Hierarchical and Distributed Model Predictive Control for Large-Scale Systems (HD-MPC), 2011. Available: http://www.ict-hd-mpc.eu/benchmarks/HD-MPC_hydropower_valley_public_benchmark.zip.

C. Savorgnan, C. Romani, A. Kozma, and M. Diehl. Multiple shooting for distributed systems with applications in hydro electricity production. *Journal of Process Control*, 21(5):738 – 745, 2011. Special Issue on Hierarchical and Distributed Model Predictive Control.

S. Sawadogo, R. Faye, P. Malaterre, and F. Mora-Camino. Decentralized predictive controller for delivery canals. In *Proceedings of 1998 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3880–3884 vol.4, San Diego, CA, 1998.

R. Scattolini. Architectures for distributed and hierarchical model predictive control - A review. *Journal of Process Control*, 19(5):723–731, May 2009.

J. Schuurmans, A. Hof, S. Dijkstra, O. H. Bosgra, and R. Brouwer. Simple water level controller for irrigation and drainage canals. *Journal of Irrigation and Drainage Engineering*, 125(4):189–195, July 1999.

D. D. Šiljak. *Large-Scale Dynamic Systems: Stability and Structure*. North Holland, New York, NY, 1978.

M. G. Singh and A. Titli. *Systems: Decomposition, Optimisation, and Control*. Pergamon, Oxford, UK, 1978.

The Mathworks. MATLAB Optimization Toolbox, 2012. http://www.mathworks.nl/products/optimization/.

K.-C. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimation*, 6:615–640, 2010.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. Submitted to SIAM Journal on Optimization. Available: http://www.math.washington.edu/~tseng/papers/apgm.pdf, May 2008.

J. N. Tsitsiklis, D. D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, Sept. 1986.

P.-J. van Overloop. *Model Predictive Control on Open Water Systems*. PhD thesis, Delft University of Technology, The Netherlands, 2006.

P.-J. van Overloop, A. J. Clemmens, R. J. Strand, R. M. J. Wagemaker, and E. Bautista. Real-time implementation of model predictive control on Maricopa-Stanfield irrigation and drainage district's WM canal. *Journal of Irrigation and Drainage Engineering*, 136(11):747–756, 2010.

A. Venkat, J. Rawlings, and S. Wright. Distributed model predictive control of large-scale systems. In R. Findeisen, F. Allgwer, and L. Biegler, editors, *Assessment and Future Directions of Nonlinear Model Predictive Control*, volume 358 of *Lecture Notes in Control and Information Sciences*, pages 591–605. Springer, Berlin / Heidelberg, Germany, 2007.

A. Venkat, I. Hiskens, J. Rawlings, and S. Wright. Distributed MPC strategies with application to power system automatic generation control. *IEEE Transactions on Control Systems Technology*, 16(6):1192–1206, Nov. 2008.

Curriculum Vitae

Minh Dang Doan was born on November 20, 1981, in Cantho, Vietnam. In 1999, he was selected to the "Programme de Formation d'Ingénieurs d'Excellence au Vietnam", and then obtained the B.Eng. degree in Mechatronics from Ho Chi Minh City University of Technology (Vietnam) in 2004.

In 2006, he received the Mekong1000 scholarship from Cantho Province to study in a master program at Delft University of Technology (The Netherlands), where he obtained the M.Sc. degree in Systems and Control in 2008.

From November 2008 to November 2012, he worked on his Ph.D. project, funded by the European Union Seventh Framework STREP project "Hierarchical and Distributed Model Predictive Control", under the supervision of professors Bart De Schutter and Tamás Keviczky. The main focus of his Ph.D. research is distributed optimization algorithms for model predictive control of large-scale systems.