UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DISTRIBUTED MATRIX ANALYSIS AND COMPUTATION OVER NETWORKS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

AMINAT BUSAYO OYELEKE
Norman, Oklahoma
2024

DISTRIBUTED MATRIX ANALYSIS AND COMPUTATION OVER NETWORKS

A THESIS APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Choon Yik Tang (Chair)

Dr.Joseph P. Havlicek

Dr. Erkan Kayacan

Acknowledgments

Profound gratitude to God for the strength, wisdom, and perseverance that guided me through this journey.

Deepest appreciation to my advisor, Dr. Choon Yik Tang, for his unwavering support, invaluable guidance, and insightful feedback that shaped this research and refined this thesis. His knowledge and encouragement have been a constant source of inspiration.

Thanks to my thesis committee, Dr. Havlicek and Dr. Kayacan, for their dedicated time and valuable critiques that enriched this work.

The University of Oklahoma provided a supportive research environment. Gratitude to faculty and staff for their guidance and support, instrumental in my academic and personal growth.

To my family, my core support system, I owe immense gratitude. My parents, Barr. Idowu Oyeleke and Mrs. Folake Oyeleke, and siblings, Tomiwa and Hayisat Oyeleke, deserve my heartfelt appreciation for their unwavering love and unwavering faith that fueled me.

My love, Samsideen Ajala, your companionship, moral support, and shared moments of relief during challenges were invaluable. Your presence brought immense joy and strength.

Gratitude to my peers and friends for their camaraderie and support, providing moments of respite and laughter amidst the demands.

Finally, thanks to those who contributed to my academic journey, including Women in Engineering (Dr. Pittenger) and Engineering Student Life (Jeff Biggerstaff). Your support, in whatever way offered, has been invaluable.

This thesis reflects the collective effort of all who supported me, both mentioned and unmentioned.

Table of Contents

List of Figures

Abstract

This thesis introduces a continuous-time distributed algorithm designed to address a range of matrix analysis and computation problems in networked systems. Focusing initially on the Local-Equation Local-Variable (LELV) problem, the algorithm enables nodes within the network to collaboratively tackle six specific challenges. These include computing least-squares solutions to linear equations, determining the minimum-norm least-squares solution, detecting solution existence, computing the Moore-Penrose inverse of a matrix and identifying full column or row rank matrices.

The algorithm, functioning as an affine, networked dynamical system, demonstrates global exponential convergence, supported by an explicit lower bound on its convergence rate. Furthermore, it offers deterministic guarantees for some problems while ensuring convergence with probability one for others.

Extending the scope to include the Local-Equation Global-Variable (LEGV) problem, this thesis provides preliminary analysis, including equilibrium point analysis and simulation of the algorithm to demonstrate convergence. While minimal in-depth exploration was conducted, these initial insights highlight the algorithm's potential applicability in addressing LEGV challenges within distributed environments.

Overall, this thesis contributes a novel continuous-time distributed algorithm with significant implications for matrix computation in networked systems. Through rigorous theoretical analysis and initial exploration, it lays the groundwork for further research and practical applications in distributed computing settings.

Chapter 1

Introduction

1.1 Background and Motivation

In recent years, the proliferation of distributed systems and networked environments has necessitated the development of algorithms that can operate efficiently across decentralized architectures. Traditional centralized computation models are increasingly inadequate for handling the volume, velocity, and variety of data generated within these complex systems. This is particularly evident in the domain of matrix analysis and computation, where the need to process and analyze data distributed across multiple nodes poses significant computational and coordination challenges.

The critical importance of consensus within networks, especially those characterized by switching topologies and time-delays, has been well-documented. Foundational research has highlighted the intricate dynamics of agent networks and underscored the necessity for algorithms that can achieve consensus and perform complex computations under these constraints. Innovations such as gossip algorithms and Zero-Gradient Sum (ZGS) algorithms have marked significant strides toward addressing these challenges, offering robust solutions for consensus in undirected networks and distributed convex optimization, respectively.

However, this thesis is motivated by the observation that despite substantial advancements in distributed computing, there remains a critical gap in the development and understanding of algorithms that are both versatile and efficient in solving a wide

array of matrix-related problems in a distributed manner. The potential for such algorithms to transform the computational landscape of networked systems drives the research presented herein.

1.2 Literature Review

A seminal exploration in [1] emphasizes the critical importance of achieving consensus within networks characterized by switching topologies and time-delays. Their work, through continuous-time analysis, highlights the essential need for a deep understanding of the real-world dynamics prevalent in agent networks. This foundational study [1] sets a significant precedent for subsequent research in distributed systems, particularly in addressing the complexities associated with achieving agreement across diverse network configurations. This foundational work is complemented by the development of gossip algorithms [2] which offer a non-gradient-based approach to achieving consensus in undirected networks with time-varying topologies. These contributions collectively highlight the critical role of network structure and the choice of algorithms in ensuring efficient consensus mechanisms. Ref [3], [4] delve into optimizing consensus processes and online convex optimization, respectively. These works underscore the role of network sparsification and time-varying constraints in enhancing algorithmic efficiency and convergence rates, highlighting the critical balance between reducing communication costs and maintaining robust consensus mechanisms across dynamic networks. Ref [5] address distributed convex optimization by proposing Zero-Gradient Sum (ZGS) algorithms for real-time networked systems, focusing on continuous-time analysis to ensure adaptability and robustness in dynamic environments. This approach is further expanded on by introducing a distributed dual averaging algorithm, linking

convergence rates to network topology and the properties of the optimization functions [6]. These studies underline the importance of considering network structure and the nonlinear nature of optimization problems in the design of distributed algorithms. Distributed estimation in wireless sensor networks (WSNs) through clustering illustrates the value of optimizing network structure for improved performance [7]. Their approach to parameter estimation, emphasizing energy efficiency and fast convergence, highlights the need for continuous-time analysis and algorithmic efficiency in real-time data processing. The transition from centralized to distributed algorithms for solving linear equations,[8] along with the scalability improvements offered by double-layered network architectures [9], demonstrates the evolving strategies to address computational and scalability challenges in distributed settings. These contributions reveal the continuous search for methods that optimize computational loads, ensure privacy, and enhance resilience to network failures or changes. Additionally, [10], [11], delve into algorithms for solving linear equations across networks, addressing the challenges of under-determined, over-determined, and uniquely solvable systems. Futhermore, [12] extend this discussion to constrained consensus and optimization, presenting algorithms that navigate the complexities of time-varying connectivity and individual agent constraints. These contributions underscore the versatility required of distributed algorithms to handle varying system determinations and constraints efficiently. Also, [13] explore the consensus problem in fractional-order singular multi-agent systems (FOSMASs), introducing a framework for achieving robust consensus despite uncertainties. This work emphasizes the importance of considering the unique challenges posed by fractional-order systems and the need for algorithms that can adapt to singularities and time-varying network topology. The works in [14], [15], and [16] extend the discourse on distributed convex optimization, particularly focusing on constraints, nonsmooth optimization problems, and networked multi-agent systems. These studies highlight

the importance of addressing convexity through innovative algorithmic approaches, such as the augmented Lagrangian and primal-dual methods, emphasizing the role of network topology in algorithm efficiency and convergence rates. Furthermore, the exploration of non-smooth optimization problems within general linear multi-agent systems underscores the complexity and necessity for methodological innovation in tackling distributed optimization challenges. On agreement problems [17] and related explorations into dynamic average consensus on synchronous communication networks [18] contribute significantly to understanding how network structures and communication patterns influence consensus achievement. These studies underscore the critical role of network topology, including directed graphs and strong connectivity, in ensuring effective distributed decision-making and coordination. The introduction of consensus-based linear and nonlinear filtering[19] and robust distributed sensor fusion by [20] enhances the understanding of distributed estimation and sensor data processing. These works emphasize the necessity of consensus mechanisms in ensuring accurate and efficient data aggregation and estimation, pivotal for applications in sensor networks and surveillance. Explorations into leaderless coordination [21] and asynchronous algorithms for solving linear systems [22] broaden the perspective on distributed system dynamics. They highlight the challenges and solutions related to time-dependent communication, synchronization, and robustness against unpredictable network dynamics, showcasing the importance of adaptable and resilient algorithmic strategies in dynamic environments. The investigation of stability in continuous-time distributed consensus algorithms [23] and the convergence properties of distributed algorithms for linear algebraic equations [24] add depth to the discourse on the fundamental properties of distributed systems. These studies provide critical insights into the conditions necessary for ensuring stable and reliable consensus, essential for the design and analysis

of distributed control and coordination mechanisms. Furthermore, [25], [26],[27] explore the dynamics of achieving consensus in multi-agent systems under conditions of switching topology and directed graphs. The innovative approaches, such as the use of delayed feedback and matrix limit definitions, highlight the evolving strategies to enhance convergence rates and adapt to network changes, emphasizing the role of network topology and communication patterns in consensus efficiency. The exploration of distributed optimization in [27], [28], [29], provides a deep dive into the complexities of solving convex optimization problems within networked environments. The introduction of methods like the transformed primal-dual method and Nesterov's accelerated approach showcases the push towards achieving faster convergence and optimizing resource allocation through innovative algorithmic solutions. The study on asynchronous algorithms [22] and the consideration of dynamic average consensus in synchronous networks [18] add valuable perspectives on handling network dynamics and asynchronous interactions. These contributions underscore the importance of designing algorithms that are robust to time-varying network conditions and can adapt to the asynchronous nature of real-world systems, highlighting the need for flexible and resilient solutions in distributed computing. On energy efficiency and network topology optimization [30] address the challenges of extending network lifetime and enhancing algorithmic efficiency. The strategic optimization of network topology for energy conservation and the acceleration of consensus mechanisms through network sparsification reflect the growing emphasis on sustainability and cost-effectiveness in network design and operation. Improvements in distributed algorithms [31], focus on refining existing solutions to achieve faster convergence, stability, and optimal solution verification. These studies reflect the ongoing effort to refine distributed computing techniques, ensuring they are adaptable to varied and complex network configurations while maintaining high efficiency and accuracy. The exploration of event-triggered and asynchronous strategies in

distributed networks, [32], and the decentralized optimization amidst communication constraints [33], points to innovative approaches in reducing computational and communication overhead. These strategies are pivotal in addressing the challenges posed by asynchronous data exchange and dynamic network memberships, ensuring that distributed systems remain efficient and robust against changing conditions. [34]and the application of consensus and cooperation mechanisms in multi-agent systems [35] offer insights into the optimization of network topology and the analysis of node centrality. These works delve into the structural aspects of networks, aiming to enhance consensus efficiency and ensure equitable resource distribution across the network, emphasizing the importance of network design in achieving efficient distributed systems. Enabling nodes in a network to compute functions of node values through distributed strategies, employing a discrete-time analysis framework [36]. The focus on linear equations and the implication of optimization approaches to minimize differences between node values enhances our understanding of consensus mechanisms in directed networks. This research contributes to the broader discourse on distributed computing by elucidating how networked systems can efficiently calculate and reach consensus on shared functions. Leveraging the Laplacian matrix, a fundamental tool in graph theory that encapsulates the network's topology. By focusing on the spectral properties of the Laplacian matrix, particularly the eigenvalues and their algebraic multiplicities, they develop a novel method for weight matrix design that ensures finite-time consensus in digraphs [37]. This methodology is groundbreaking as it provides a systematic way to overcome the inherent challenges posed by the asymmetry of directed networks. We can tackle the challenge of optimizing a sum of convex objective functions distributed across a network through a stochastic subgradient descent approach [38]. By addressing the stochastic nature of gradients and employing weighted averaging of SSD iterates, [38] advances our knowledge of convex optimization under distributed network

constraints. The utilization of undirected graphs as communication models underpins the decentralized execution of the SSD algorithm, showcasing a methodological innovation in distributed optimization. Investigating the coordination of mobile autonomous agents through nearest neighbor rules, showcasing how local interactions can lead to emergent behavior without centralized control[39]. Their discrete-time analysis and use of switched linear systems for stability and convergence analysis provide a foundational framework for understanding the dynamics of decentralized systems. The application of graph theoretic techniques underscores the role of simple, undirected graphs in capturing the structural dynamics of agent coordination, highlighting the potential for designing distributed control algorithms in autonomous agent networks. A study emphasized achieving system determinacy through this distributed stopping mechanism, effectively navigating the complexities inherent in digraphs where unidirectional information flow can complicate consensus achievement. By employing linear iterative processes, the research facilitates a structured approach to updating node values based on weighted averages of neighboring nodes' information [40]. This methodological innovation not only simplifies the consensus process but also ensures its deterministic conclusion in a distributed setting.

The comprehensive examination of literature on matrix operations, distributed computing frameworks, and network theory establishes a solid foundation for the development of distributed computational methodologies. These seminal contributions provide deep insights into the challenges of conducting matrix analyses in complex networked systems, optimizing computational tasks across distributed entities, and achieving adaptability and efficiency in dynamic and potentially asynchronous settings. However, the focus of this thesis marks a significant expansion from these foundational efforts, diverging in several essential aspects.

First, whereas the literature often addresses consensus, optimization, and linear systems solving within narrower scopes, this body of work aims to encompass a broader spectrum of matrix computations and analyses. This not only includes the Moore-Penrose inverse but extends to encompass eigenvalue decomposition, matrix factorization, rank detection, and other critical matrix operations crucial in distributed settings. Such operations present unique computational challenges when the processing and data are distributed across multiple nodes.

Second, the prior studies mainly focus on optimization problems, consensus mechanisms, and solving linear equations, without fully engaging with the array of algorithms required for comprehensive distributed matrix computation. This body of work aims to fill this gap by developing and analyzing algorithms robust and versatile enough for the varied mathematical properties and computational demands of different matrix operations. Efforts include addressing data distribution complexities, ensuring algorithmic convergence, and optimizing network-wide computational efforts—areas not thoroughly explored in previous research.

Furthermore, this exploration delves into the impact of network dynamics on distributed matrix computations, investigating how topology, communication delays, and agent failures affect the computations' accuracy, stability, and efficiency. This focus brings to light additional complexities compared to the general discussions of consensus and optimization found in existing literature, necessitating maintaining computational integrity amid changing network conditions and adapting algorithms to the network's structural nuances.

In conclusion, the existing literature provides a foundational backdrop for distributed computations and network dynamics understanding. However, the exploration of this thesis ventures into broader territories, addressing a comprehensive range of computational challenges across distributed matrix operations. By introducing novel

algorithmic approaches tailored to the diverse needs of matrix analysis in distributed networks, this work significantly contributes to the knowledge base, paving the way for further advancements and applications in distributed computing and matrix theory.

1.3 Thesis Contributions

This thesis introduces a novel continuous-time distributed algorithm designed to empower nodes within a network to collaboratively solve six specific matrix analysis and computation problems, each characterized by distinct subsets of problem data known only to individual nodes. The versatility of this algorithm is demonstrated through its application to a diverse set of problems, including but not limited to finding least-squares solutions to systems of linear equations, computing the Moore-Penrose inverse of a matrix, and determining the rank of a matrix.

A key innovation of this work is the algorithm's ability to operate as an affine, networked dynamical system, which is shown to be globally exponentially convergent. An explicit lower bound on its convergence rate is derived, underscoring the algorithm's efficiency and the speed with which it can achieve consensus across a range of matrix computation tasks. Furthermore, the algorithm distinguishes itself by offering deterministic guarantees for certain problems and probabilistic assurances for others, showcasing its adaptability and robustness in dynamic network environments.

The implications of this research are far-reaching, offering a scalable and efficient framework for this work that can be seamlessly integrated into various network configurations. This work not only addresses a critical gap in the literature but also sets a precedent for future research in distributed computing, paving the way for the development of more advanced algorithms capable of tackling an even broader spectrum of computational challenges in distributed settings.

1.4 Thesis Outline

This thesis explores the intersection of distributed computing and matrix analysis. The opening chapter establishes the context by highlighting recent advancements in distributed computing and emphasizing the critical role of matrix analysis in networked systems. It then introduces the core contribution: a novel continuous-time distributed algorithm designed to tackle six specific problems related to matrix analysis and computation. The chapter outlines the research objectives and presents the overall structure of the thesis. It delves into existing literature to identify critical gaps that necessitate a new algorithm, thereby establishing the significance of the proposed solution.

The subsequent chapter, Chapter 2 dives deep into the theoretical foundation of the Local-Equation Local-Variable (LELV) approach for the distributed computing model used in the thesis. It outlines the underlying assumptions and problem formulations. Six distinct matrix analysis and computation problems are meticulously detailed, providing a framework for understanding the algorithm's functionality. The chapter then delves into the heart of the research, the design and operational logic of the proposed algorithm. It presents mathematical derivations and proofs demonstrating the algorithm's ability to solve the defined problems with guaranteed deterministic solutions. Furthermore, the chapter analyzes the algorithm to ensure its reliability across various network conditions and time. It distinguishes between problems solved with deterministic guarantees and those addressed probabilistically. The chapter concludes by exploring potential extensions of the algorithm, hinting at its future development and broader applications in distributed computing.

Chapter 3 introduces a novel algorithm specifically designed to address the challenges associated with the Local-Equation Global-Variable (LEGV) problem within distributed computing environments. It emphasizes the significance of this problem

and formally defines its mathematical and computational framework. Focusing on the algorithm design, the chapter details its scalability and adaptability to network dynamics. A preliminary analysis is then conducted to evaluate the algorithm's performance, setting the stage for a more in-depth exploration in subsequent chapters.

The final chapters document the practical implementation of both the LELV and LEGV algorithms. It details the computational setup, simulation designs, and chosen parameters. The summary and results are then meticulously analyzed to demonstrate the effectiveness of the algorithms in solving the targeted matrix analysis problems, showcasing their performance and robustness mentioned in the conclusion.

Chapter 2

Local-Equation Local-Variable Problem

2.1 Introduction

This chapter delves into the "Local-Equation Local-Variable Problem" within the context of this thesis. The section introduces the foundational concepts and motivations behind the study, setting the stage for a detailed exploration of the problem formulation, algorithmic solutions, and comprehensive analyses.

2.2 Problem Formulation

In the pursuit of embarking on the exploration of the current work, tackling the challenge of efficiently and robustly performing a variety of matrix operations. These operations, crucial across various fields such as signal processing and linear systems resolution, present unique challenges in the context of network communications.

The broad domain of this work within network systems confronts substantial challenges. It involves not just the computation of Moore-Penrose inverses (MPI) but extends to a wide array of matrix operations essential across numerous applications. This exploration is rooted in a control-theoretic framework, employing a dynamical system of differential equations to capture the distributed computations across a network.

The core of this investigation is articulated through a dynamical system model, represented as:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}, \tag{2.1}$$

where $\dot{\mathbf{x}}(t) = [x(t); y(t); z(t); w(t)]^\top$ denotes the state vector, encompassing the distributed computational state across the network nodes at time t. The dynamics of the system are governed by:

$$\mathbf{A} = \begin{bmatrix} 0 & A^\top & 0 & 0 \\ -A & -I_m & 0 & 0 \\ 0 & 0 & -I_n & A^\top \\ A & 0 & -A & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ b \\ 0 \\ 0 \end{bmatrix}. \tag{2.2}$$

One primary aspect of this investigation focuses on a control-theoretic model represented by a dynamical system of differential equations. Here, A represents the matrix to be analysed, $A^\top$ denotes its transpose, and I_m is the identity matrix of appropriate dimensions. The vector b introduces an external input to the system, highlighting its interaction with the environment.

Pros: The formulation's strength lies in its ability to explicitly model the distributed nature of the problem, where each component of $\mathbf{x}(t)$—namely, $x(t)$, $y(t)$, $z(t)$, and $w(t)$—has distinct physical meanings, representing what each node in the network computes or knows.

Cons: A notable drawback of this formulation is the opaque insight into the system's internal dynamics, particularly how the network nodes interact and converge.

Moreover, the presence of the forced term $\mathbf{b}$ adds a layer of complexity, necessitating further analysis to understand its influence on the system's behavior and stability. This model serves as a foundation for addressing a broad spectrum of matrix-related problems within distributed networks.

Given the above, the goal of this paper is to design a continuous-time distributed algorithm that enables the N nodes to cooperatively solve two types of matrix analysis and computation problems. To define these two types of problems, let $\mathbf{x}^* = \{z \in \mathbb{R}^n : \|Az - b\| = \min_{x\in\mathbb{R}^n} \|Ax - b\|\}$ denote the set of least-squares solutions to (1), $R(A) \subset \mathbb{R}^m$ denote the range of A, $N(A) \subset \mathbb{R}^n$ denote the nullspace of A, and $R(A)^\perp \subset \mathbb{R}^m$ denote the orthogonal complement of the range of A. The first type is concerned with the system of linear equations (1) and contains the following three problems:

(P1) Each node $i \in \mathcal{V}$ wishes to find $x_i^* \in \mathbb{R}^{n_i}$ such that $x^* = [x_1^{*T} x_2^{*T} \cdots x_N^{*T}]^T \in \mathbb{R}^n$ is a least-squares solution to (1), i.e., $x^* \in X^*$.

(P2) Each node $i \in \mathcal{V}$ wishes to find $z_i^* \in \mathbb{R}^{n_i}$ such that $z^* = [z_1^{*T} z_2^{*T} \cdots z_N^{*T}]^T \in \mathbb{R}^n$ is the minimum-norm least-squares solution to (1), i.e., $z^* \in X^*$ and $\|z^*\| \leq \|x^*\|$ for all $x^* \in X^*$.

(P3) Each node $i \in \mathcal{V}$ wishes to detect whether a solution to (1) exists, i.e., whether $b \in R(A)$.

The second type is concerned only with matrix A in (1) and contains the following three problems:

(P4) Each node $i \in \mathcal{V}$ wishes to compute $A_i^\dagger \in \mathbb{R}^{n_i \times m}$ such that $A^\dagger = [A_1^{\dagger T} A_2^{\dagger T} \cdots A_N^{\dagger T}]^T \in \mathbb{R}^{n\times m}$ is the Moore-Penrose inverse of A.

(P5) Each node $i \in \mathcal{V}$ wishes to detect whether A has full column rank, i.e., whether $\text{rank}(A) = n$.

(P6) Each node $i \in \mathcal{V}$ wishes to detect whether A has full row rank, i.e., whether $\text{rank}(A) = m$.

2.3 Algorithm Description

In distributed systems or computations, the LELV paradigm facilitates the division of a global problem into smaller, manageable sub-problems that can be solved independently by individual nodes or agents within the network. This approach is visually represented in Figure (2.1), which abstractly illustrates the notion of local computations contributing to a global solution.

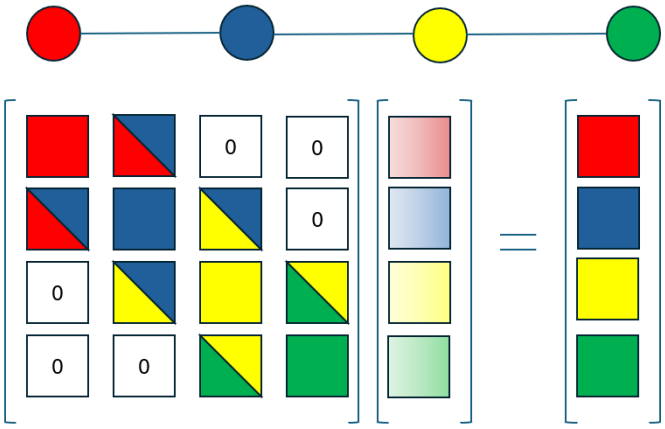

Figure 2.1: The Local-Equation Local Variable Problem.

The colored circles in the illustration symbolize distinct nodes in a network, each tasked with a specific segment of the overarching computational endeavor. The colored, diagonally divided blocks may be interpreted as matrices or data arrays, which encapsulate the unique set of data that each node processes—termed as 'local data'.

Notably, the presence of zero-filled blocks indicates segments of the entire dataset that are either irrelevant to particular nodes or simply not within their data jurisdiction.

Crucially, each node employs a local equation to resolve its allocated portion of the grand problem, operating exclusively on local variables represented by the non-zero elements of the colored blocks. Upon completion of these independent computations, the individual solutions can be coalesced—signified by the equal sign transitioning into aggregated blocks—to reconstruct the complete matrix or deduce the global solution to the posed problem.

In the realm of numerical linear algebra, for instance, this could be akin to each node calculating the solution to a local linear system $A_i x_i = b_i$. Here, A_i embodies a submatrix and x_i, b_i respectively stand for local portions of the ultimate solution and right-hand side vectors. The conclusive solution x to the global system $Ax = b$ is ascertained by the synthesis of all local solutions x_i.

Suppose each node $i \in \mathcal{V}$ updates these four state variables according to

$$\dot{x}_i(t) = \sum_{j \in \mathcal{N}^i} A_{ji}^T y_j(t), \tag{2.3}$$

$$\dot{y}_i(t) = -\sum_{j \in \mathcal{N}^i} A_{ij} x_j(t) - y_i(t) + b_i, \tag{2.4}$$

$$\dot{z}_i(t) = -z_i(t) + \sum_{j \in \mathcal{N}^i} A_{ji}^T w_j(t), \tag{2.5}$$

$$\dot{w}_i(t) = \sum_{j \in \mathcal{N}^i} A_{ij} x_j(t) - \sum_{j \in \mathcal{N}^i} A_{ij} z_j(t), \tag{2.6}$$

Consider the dynamical system given in matrix form by the differential equation:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \\ \dot{w}(t) \end{bmatrix} = \begin{bmatrix} 0 & A^\top & 0 & 0 \\ -A & -I_m & 0 & 0 \\ 0 & 0 & -I_n & A^\top \\ A & 0 & -A & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ w(t) \end{bmatrix} + \begin{bmatrix} 0 \\ b \\ 0 \\ 0 \end{bmatrix}, \tag{2.7}$$

where, $[x(t); y(t); z(t); w(t)]$ represents the state vector of the system at time t, and b is an external input vector to the system.

Given the differential equation system for a network of nodes, the aim is to compute distributed matrix operations across the network. For this approach, a simplified scenario with three nodes to develop and understand our approach, which can then be generalized to an i-node system. In the realm of network science and engineering, the conceptualization and analysis of networks as mathematical graphs provide a foundational framework for understanding complex systems. This thesis delves into the study of such networks, particularly focusing on those modeled as undirected, connected graphs. Formally, we denote such a network by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where:

- $\mathcal{V} = \{1, 2, \ldots, N\}$ represents the set of nodes within the network, encompassing a total of $N \geq 2$ distinct entities. These nodes could symbolize various components depending on the network's nature, ranging from routers in a communication network to individuals in a social network.

- $\mathcal{E} \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ constitutes the set of edges linking these nodes. An edge $\{i, j\}$ signifies a bidirectional, unweighted relationship between nodes i and j, allowing for the flow of information or resources.

Crucially, any two nodes $i, j \in \mathcal{V}$ are considered neighbors with the capability to communicate directly if and only if $\{i, j\} \in \mathcal{E}$. This neighborly relation is pivotal, as it underlines the network's connectivity and potential pathways for communication or data transfer. The neighborhood of each node $i \in \mathcal{V}$, denoted by $\mathcal{N}_i = \{j \in \mathcal{V} : \{i, j\} \in \mathcal{E}\}$, encapsulates all direct connections of i, influencing the node's role and significance within the network.

For the purpose of this analysis, we assume that communications between neighboring nodes are instantaneous, delay-free, and devoid of errors. This idealized assumption allows us to focus on the topological and structural aspects of the network, sidestepping the complexities introduced by real-world communication imperfections. Such a model serves as a valuable abstraction, enabling the investigation of fundamental properties and behaviors intrinsic to the network's design. The study of network dynamics and their intrinsic mathematical representations has increasingly gained prominence, particularly in the context of understanding how complex network systems can solve collective computational problems. A quintessential aspect of this exploration is the association of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a system of linear equations, denoted by $Ax = b$. This system not only encapsulates the computational tasks distributed across the network but also mirrors the structural complexity and interconnectedness inherent to $\mathcal{G}$. The equation can be represented as:

$$Ax = b, \tag{2.8}$$

where the matrix $A \in \mathbb{R}^{m \times n}$ (with the precondition that $A \neq 0$ to avoid trivial solutions) and vectors $x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ conform to a block partitioning that reflects the network's topology.

In this formulation, each block $A_{ij} \in \mathbb{R}^{m_i \times n_j} \, \forall i, j \in \mathcal{V}$, with $x_i \in \mathbb{R}^{n_i}$ and $b_i \in \mathbb{R}^{m_i} \, \forall i \in \mathcal{V}$, signifies the localized information or data known exclusively to specific nodes or shared between directly connected neighbors. This granularity allows the network to collectively address the computational challenge posed by $Ax = b$ without necessitating centralized data pooling or processing, thereby adhering to the decentralized nature of $\mathcal{G}$. The dimensions m and n aggregate the individual contributions of all nodes, expressed as:

$$m = \sum_{i \in \mathcal{V}} m_i, \quad n = \sum_{i \in \mathcal{V}} n_i, \tag{2.9}$$

elucidating the cumulative scale of the system in terms of the network size and the distribution of computational responsibilities.

Importantly, the distribution of A_{ij} and b_i across the nodes is meticulously designed to mirror the network's communication paths, ensuring that each node possesses only a subset of the overall data, specifically:

- For every node i, the information regarding A_{ii} and b_i remains confined to i, underscoring the localized knowledge principle critical for distributed computing.
- For any two distinct nodes i and j, the block A_{ij} is accessible either to i or j if there exists an edge $\{i, j\} \in \mathcal{E}$, embodying the edge-induced sharing mechanism. Conversely, A_{ij} is set to 0 in the absence of a direct link, aligning with the graph's adjacency structure.

This intricate setup not only resonates with the concept of weighted adjacency and Laplacian matrices, typically employed to encode graph connectivity, but also extends the scope to more generalized data representations within network matrices. However, a notable constraint emerges from the reluctance or inability of nodes to divulge their A_{ij}'s and b_i's, attributed to security, privacy, or bandwidth considerations. Such restrictions necessitate innovative approaches to distributed problem-solving, wherein

nodes collaborate to deduce global solutions from their local data without explicit data exchanges that could compromise the system's integrity or operational efficiency.

Through this elaborate construct, the study aims to unravel methodologies and algorithms capable of navigating the dual challenges of distributed data processing and stringent communication constraints, ultimately fostering advancements in network-driven computational paradigms.

By dissecting the intricate web of interactions and relationships, we can uncover insights into network resilience, efficiency, and capacity for information dissemination, all of which are crucial for the design and optimization of contemporary and future networked systems.

2.4 Algorithm Analysis

The distributed algorithm's design incorporates the dynamics of a specific matrix A, facilitating efficient computation across the network. The structure of A suggests a system with interconnected state variables, guiding the algorithm's local computation and communication steps.

2.4.1 SVD Basics

The full Singular Value Decomposition (SVD) offers an elegant methos for analyzing matrices and solving systems of linear equations, including those that are underdetermined or overdetermined. For matrix $A \in \mathbb{R}^{m \times n}$, SVD decomposes A as:

$$A = U\Sigma V^T$$

where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $\Sigma \in \mathbb{R}^{m \times n}$ is a matrix that contains the positive singular values of A. The Moore-Penrose inverse of A, denoted as $A^{\dagger}$, can then be computed as:

$$A^{\dagger} = V\Sigma^{+}U^{T}$$

where Σ^{+} is obtained by taking the reciprocal of each non-zero element on the diagonal of Σ, and transposing the matrix.

In this specific application, decomposition allows for the distributed computation of the Moore-Penrose inverse by distributing parts of U, Σ, and V across the network nodes. For a network of three nodes, each node is assigned a portion of the computation related to U, Σ, and V. This setup models the initial case and provides insight into how the approach scales with the number of nodes.

Generalizing to an i-node system, partitioning the computation such that each node is responsible for a portion of the SVD computation, facilitating the distributed computation. This method leverages the network's distributed architecture to efficiently compute the inverse, essential for solving the system of linear equations represented by the initial differential equation system.

The efficacy of this approach in a distributed environment hinges on the network's ability to efficiently share and aggregate the computations performed by individual nodes, highlighting the interplay between network topology and computational efficiency.

Continuing exploration of the distributed computation of Moore-Penrose inverses over networks, a critical step involves transforming the original system to reveal its full dynamics and facilitate analytical insights. To achieve this, we apply a transformation

that diagonalizes the model, enabling a clearer analysis of convergence behaviors and equilibrium points. Specifically, the transformed system is given by:

$$\dot{\tilde{\mathbf{x}}}(t) = T^{-1}AT\tilde{\mathbf{x}}(t) + T^{-1}b, \tag{2.10}$$

where $\tilde{\mathbf{x}}(t) = T^{-1}\mathbf{x}(t)$ represents the state vector in the transformed space, and T is the transformation matrix that diagonalizes A.

This transformation yields a system in the form $\dot{\tilde{\mathbf{x}}}(t) = \tilde{\mathbf{A}}\tilde{\mathbf{x}}(t) + \tilde{\mathbf{b}}$, where $\tilde{\mathbf{A}} = T^{-1}\mathbf{A}T$ and $\tilde{\mathbf{b}} = T^{-1}b$. This new formulation offers a comprehensive view into the system's dynamics, providing a platform for rigorous analysis of convergence rates and the identification of equilibrium points.

Advantages:

- The diagonalized form $\tilde{\mathbf{A}}$ allows for a straightforward analysis of the system's stability and convergence properties, making it possible to derive explicit conditions under which the system will converge to the desired Moore-Penrose inverse.

- It facilitates a detailed examination of the system's equilibrium points, offering insights into the long-term behavior of the networked agents.

Limitations:

- One significant limitation of this approach is the loss of the direct physical interpretation of the state variables in the transformed system. The variables in $\tilde{\mathbf{x}}(t)$ may not correspond to quantities that each node in the network can directly compute or interpret.

- Additionally, the presence of the forced term $\tilde{\mathbf{b}}$ in the transformed system still poses challenges, requiring careful consideration in the analysis to ensure that its influence does not hinder achieving consensus or introduce instability.

Incorporating this transformation into the problem formulation deepens the understanding of the system's dynamics and paves the way for developing distributed algorithms that are both efficient and robust. It underscores the trade-offs between gaining analytical insights and retaining the physical meaning of the system's variables, highlighting the complexities inherent in the distributed computation.

2.4.2 Equilibrium Point Analysis

Following the mathematical modeling of this system, the next step was to identify its equilibrium points. Equilibrium points play a vital role in understanding the steady-state dynamics of the system, characterized by the cessation of state variable changes over time.

The equilibrium points are determined by setting the time derivatives of the state variables to zero in the system's differential equations, resulting in a set of algebraic equations as shown in (2.7) where A represents the dynamics of the system, I is the identity matrix, and b is an external input vector. The state variables of the system are denoted by x, y, z, and w.

By setting $\dot{x}$, $\dot{y}$, $\dot{z}$, and $\dot{w}$ to zero, we derive the following equilibrium equations:

$$0 = A^T y, \tag{2.11}$$

$$0 = -Ax - y + b, \tag{2.12}$$

$$0 = -z + A^T w, \tag{2.13}$$

$$0 = Ax - Az. \tag{2.14}$$

The derived equilibrium equations provide significant insights into the system's behavior:

- Equation (2.11): $0 = A^T y$

  This equation implies that the vector y is orthogonal to the range of A^T, which means y lies in the null space of A. In the context of least squares problems, y represents the error or residual vector, orthogonal to the column space of A.

- Equation (2.12): $0 = -Ax - y + b$

  Rearranging gives $Ax + y = b$. This equation states that Ax approximates b, with y capturing the residual. This setup is typical in least squares solutions where Ax is the projection of b onto the column space of A.

- Equation (2.13): $0 = -z + A^T w$

  Rearranging to $z = A^T w$, suggests that z is determined by how the vector w (Lagrange multipliers) interacts with A^T.

- Equation (2.14): $0 = Ax - Az$

  This equation indicates that the transformations of x and z by A are identical, meaning $x - z$ is in the kernel of A. This allows for an infinite number of solutions if A is not of full rank, highlighting underdetermined characteristics of the system.

Analysis of Equilibrium Points and System Solutions

Given the system of equations $Ax = b$, where the range of A is defined as the span of A's columns, we explore the conditions under which solutions exist and the nature of these solutions from the perspective of optimization and error minimization.

Understanding the Equilibrium Points

The fundamental premise in linear algebra that a solution x such that $Ax = b$ exists if and only if b is within the range of A, guides the analysis of the system's equilibrium points:

1. The condition $Ax = b$ indicates that, at equilibrium, for any b in the range of A, there exists at least one solution x. This confirms the system's ability to reach a state of equilibrium given the appropriate conditions.

2. The variable y, representing minimized error, is orthogonal to the columns of A. This signifies that y captures the component of b not representable by the column space of A, characteristic of the least-squares solution.

3. The minimization criterion $\|Ax - b\|$ suggests that x represents a least-squares solution, optimizing the approximation of b through A's column space when an exact solution is unattainable.

4. The similarity in the role of z indicates the presence of multiple solutions that minimize the representation error, pointing to the least-squares solution's nature under certain conditions.

5. The variable w, interpreted as a Lagrange multiplier, reflects the system's sensitivity to changes in b, providing insights into the constraints' impact on the solution.

Derivation from Least-Squares Interpretation to $A^T y = 0$

The least-squares method minimizes the sum of squared residuals, aiming to solve the optimization problem:

$$\min_x \|Ax - b\|^2. \tag{2.15}$$

The residuals in this context are defined as $y = b - Ax$. The optimization leads us to the normal equations:

$$A^T Ax = A^T b, \tag{2.16}$$

obtained by setting the gradient of the objective function with respect to x to zero. This equation is pivotal in finding the least-squares solution.

An inherent aspect of the least-squares solution is the orthogonality principle, which states that the residuals y are orthogonal to the column space of A. Mathematically, this translates to the condition:

$$A^T y = 0. \tag{2.17}$$

This condition emerges because the dot product of y with any vector in the column space of A equals zero, indicating that y lies in the left null space of A. The equation $A^T y = 0$ succinctly captures this relationship, highlighting the orthogonality of the residuals to the space spanned by the columns of A.

Explanation of $A^T y = 0$ in Terms of Orthogonality

The equation $A^T y = 0$ plays a significant role in the geometric interpretation of the least-squares method. It suggests that the residuals vector y, which is the difference between the observed values b and the predictions Ax, is orthogonal, or perpendicular, to the range space of A. Here, we delve into the implications of this orthogonality.

Geometric Interpretation

The columns of the matrix A define its range space, which represents all possible linear combinations of these columns. The condition $A^T y = 0$ implies that the dot product of the residuals vector y with any vector in the range space of A is zero. This can be expressed as:

$$A^T y = 0 \quad \Longrightarrow \quad \text{for all } v \in \mathcal{R}(A), \quad y^T v = 0, \tag{2.18}$$

where $\mathcal{R}(A)$ denotes the column space of A. This condition signifies that y is perpendicular to every vector in the range space of A, or equivalently, y is orthogonal to the column space of A.

Implications of Orthogonality

This orthogonality principle is fundamental to the least-squares solution, indicating that the best approximation of b by Ax results in a residuals vector $y = b - Ax$ that lies in a direction entirely orthogonal to the space spaned by the columns of A. Therefore, the equation $A^T y = 0$ encapsulates the essence of the least-squares method, highlighting that the residuals are not only minimized in magnitude but also geometrically orthogonal to the predictive space defined by A.

Derivation of the Least-Squares Problem from the Given Equation

The equation presented,

$$0 = -Ax - y + b, \tag{2.19}$$

can be rearranged to highlight its connection to a least-squares formulation. By reorganizing the equation, we obtain:

$$Ax + y = b. \tag{2.20}$$

This equation indicates that y, the residual vector, is the difference between the actual outcomes b and the predictions made by the linear model Ax. The least-squares method seeks to minimize these residuals, thus optimizing the fit between the model's predictions and the actual data.

Formulating the Least-Squares Problem

The essence of the least-squares problem is to find the parameter vector x that minimizes the sum of squared residuals. This objective can be mathematically formulated as:

$$\min_x \|Ax - b\|^2. \tag{2.21}$$

This minimization problem focuses on reducing the squared Euclidean norm of the residual vector $y = b - Ax$. The squared norm $\|y\|^2 = \|Ax - b\|^2$ represents the sum of the squares of the individual components of y, encapsulating the total error between the model's predictions and the actual observations.

Connecting to the Original Equation

From the initial equation, $Ax_ls + y = b$, we see that minimizing $\|Ax - b\|^2$ is equivalent to minimizing $\|y\|^2$, as y embodies the deviation of Ax from b. Thus, the process of minimizing the residuals directly corresponds to addressing the least-squares problem, aiming to find an optimal x that achieves this minimization. Therefore, the equation $0 = -Ax - y + b$ inherently leads to a least-squares problem where the goal is to minimize the squared norm of the residuals, $\|Ax - b\|^2$.

Minimum-Norm Interpretation of the Equilibrium Point $0 = Ax - Az$

The equilibrium condition $0 = Ax - Az$ implies that the vectors x and z, upon transformation by the matrix A, result in identical outcomes. This can be seen as:

$$Ax = Az, \tag{2.22}$$

which further simplifies to indicate that the difference between x and z lies within the null space of A, assuming A is nonzero.

Minimum-Norm Solution

In scenarios where multiple solutions exist, the minimum-norm solution is sought to find the vector with the smallest Euclidean norm that satisfies the equation. Applying this concept to this condition, if x is a known state vector, then selecting a z that minimizes the deviation from x, under the constraint $Ax = Az$, aligns with seeking a minimum-norm solution.

This interpretation emphasizes not just the existence of multiple solutions but the preference for the solution with the least magnitude, particularly relevant in underdetermined systems or when optimizing for stability and efficiency in solutions. It highlights the importance of considering the norm of potential solutions in the context of linear equations and system dynamics.

Interpretation and Derivation of Equilibrium from the Lagrangian

Given the Lagrangian formulation:

$$\mathcal{L}(z, w) = \frac{1}{2}\|z\|^2 + w^T(Ax_{ls} - Az),$$

where w^T denotes the vector of Lagrange multipliers, x_{ls} the least-squares solution of $Ax = b$, and $\frac{1}{2}\|z\|^2$ aims to minimize the Euclidean norm of z, we explore the optimization problem's equilibrium conditions.

Optimization Conditions

- Differentiating with respect to z and setting the derivative equal to zero yields:

$$\nabla_z \mathcal{L} = z - A^T w = 0 \implies z = A^T w.$$

  This indicates the optimal z in terms of the Lagrange multipliers, adhering to the minimum-norm principle within the constrained optimization framework.

- Differentiation with respect to w enforces the constraints:

$$\nabla_w \mathcal{L} = Ax_{ls} - Az = 0,$$

  ensuring the solution respects the deviation of Az from the desired projection Ax_{ls}, effectively highlighting the adherence to the least-squares solution constraints.

2.4.3 Coordinate Transformation

The transformation to the $\tilde{x}$ domain using the SVD components enables a clearer understanding of the system's behavior and simplifies the control problem. Specifically, the diagonalization through Σ highlights the dominant dynamics of the system, and the partitioning of $\mathbf{A}'$ into blocks with Σ, Σ^T, and $-\mathbf{I}$ allows for an isolated analysis of each dynamic component.

Furthermore, the introduction of $\tilde{b}$ as a transformation of the original input vector $\mathbf{b}$ by U^{-1} aligns the external inputs with the transformed system states, ensuring that the control strategies designed in the $\tilde{x}$ domain are coherent and applicable to the original system when transformed back. To find the transformed matrix $\tilde{\mathbf{A}} = T^{-1}\mathbf{A}T$, we consider the given matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} 0 & A^T & 0 & 0 \\ -A & -I_m & 0 & 0 \\ 0 & 0 & -I_n & A^T \\ A & 0 & -A & 0 \end{bmatrix}$$

and the transformation matrix T defined as:

$$T = \begin{bmatrix} V & 0 & 0 & 0 \\ 0 & U & 0 & 0 \\ 0 & 0 & V & 0 \\ 0 & 0 & 0 & U \end{bmatrix}$$

Given T, its inverse T^{-1} is the transpose of T because T is orthogonal:

$$T^{-1} = T^T = \begin{bmatrix} V^T & 0 & 0 & 0 \\ 0 & U^T & 0 & 0 \\ 0 & 0 & V^T & 0 \\ 0 & 0 & 0 & U^T \end{bmatrix}$$

Applying the transformation $\tilde{\mathbf{A}} = T^{-1}\mathbf{A}T$, we aim to simplify or understand the structure of $\mathbf{A}$ in a new basis defined by T. The transformed matrix $\tilde{\mathbf{A}}$ is then calculated as follows:

$$\tilde{\mathbf{A}} = T^{-1}\mathbf{A}T = \begin{bmatrix} 0 & U^T A^T V^T & 0 & 0 \\ -AVU^T & -UU^T & 0 & 0 \\ 0 & 0 & -VV^T & UA^T V^T \\ AVU^T & 0 & -AVU^T & 0 \end{bmatrix}$$

Substituting $A = V\Sigma U^T$ into the transformed matrix and considering the properties of U, V, and Σ, we obtain:

The overall system order in the transformed domain is $2n + 2m$, indicating the comprehensive state representation that includes both the original states and additional states introduced by the transformation. Given the original system dynamics represented by the matrix $\mathbf{A}$ and a vector $\mathbf{b}$, we apply a transformation to simplify the system's representation for analysis. The transformed system dynamics are characterized by the equation:

$$\dot{\tilde{x}} = \tilde{\mathbf{A}}\tilde{x} + \tilde{\mathbf{b}}$$

where

$$\tilde{\mathbf{A}} = \begin{bmatrix} 0 & \Sigma^T & 0 & 0 \\ -\Sigma & -I_m & 0 & 0 \\ 0 & 0 & I_n & \Sigma^T \\ \Sigma & 0 & -\Sigma & 0 \end{bmatrix}, \qquad \tilde{\mathbf{b}} = \begin{bmatrix} 0 \\ \tilde{b} \\ 0 \\ 0 \end{bmatrix}$$

where, $\tilde{\mathbf{b}} = U^{-1}b$. The matrix Σ represents the singular values from the Singular Value Decomposition of a component matrix of $\mathbf{A}$, and I is the identity matrix matching the dimensions of Σ. The transformation employs matrices U and V, deriving from the SVD, to facilitate the analysis and control design of the system by simplifying its dynamics. The final equilibrium point dynamics is given by:

$$\begin{bmatrix} \dot{\tilde{x}}_i \\ \dot{\tilde{y}}_i \\ \dot{\tilde{z}}_i \\ \dot{\tilde{w}}_i \end{bmatrix} = \left[\begin{array}{cc|cc} 0 & \sigma_i & 0 & 0 \\ -\sigma_i & -1 & 0 & 0 \\ \hline 0 & 0 & -1 & \sigma_i \\ \sigma_i & 0 & -\sigma_i & 0 \end{array}\right] \begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ \tilde{z}_i \\ \tilde{w}_i \end{bmatrix} + \begin{bmatrix} 0 \\ \tilde{b}_i \\ 0 \\ 0 \end{bmatrix} \tag{2.23}$$

Dynamics of Additional System Components

In addition to the primary dynamics facilitated by the singular values indexed by $i = 1, 2, \ldots, r$, the system's higher order components, indexed by $i = r+1, \ldots, n$

and $i = r+1, \ldots, m$, exhibit specific behaviors based on the absence of corresponding singular values in these ranges. These components are critical for understanding the full system dynamics in the transformed domain, particularly in scenarios where the system is over-specified or under-specified.

Dynamics for $i = r+1, \ldots, n$, i.e., for the components beyond the rank r of the matrix A:

- The states $\tilde{x}_i$ are static since they correspond to null singular values, implying:

$$\dot{\tilde{x}}_i = 0 \implies \tilde{x}_i(t) = \text{const} = \tilde{x}_i(0).$$

- For the z components, the dynamics show exponential decay:

$$\dot{\tilde{z}}_i = -\tilde{z}_i \implies \tilde{z}_i(t) = e^{-t}\tilde{z}_i(0),$$

  reflecting the system's ability to stabilize these components naturally over time.

Dynamics for $i = r+1, \ldots, m$, i.e., for the additional components in the transformed domain, which may exceed the rank r but are within the dimension m of A:

- The y components are influenced both by decay and external inputs:

$$\dot{\tilde{y}}_i = -\tilde{y}_i + \tilde{b}_i \implies \tilde{y}_i(t) = e^{-t}\tilde{y}_i(0) + \tilde{b}_i,$$

  which indicates a dynamic settling towards a state influenced by the transformed input $\tilde{b}_i$.

- The w components remain constant throughout:

$$\dot{\tilde{w}}_i = 0 \implies \tilde{w}_i(t) = \text{const} = \tilde{w}_i(0).$$

2.4.4 Problem P1: Least-Squares Solution

Problem P1 entails each node finding a component of the least-squares solution to $A\mathbf{x} = \mathbf{b}$. Through distributed computation, nodes iteratively refine their state variables to converge towards the global least-squares solution, leveraging consensus-based algorithms to minimize the collective residual error.

Theorem 1. *For any initial states $x(0), y(0), z(0)$, and $w(0)$, the distributed algorithm ensures that $x(t)$ converges to a least-squares solution, $x^* \in \mathcal{X}^*$, such that x^* minimizes $\|Ax - b\|^2$.*

Proof for Problem P1: Finding the Least-Squares Solution

Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ and a vector $b \in \mathbb{R}^m$, the least-squares problem aims to minimize $\|Ax - b\|^2$ over $x \in \mathbb{R}^n$. Utilizing the Singular Value Decomposition (SVD) of A, we express A as $A = U\Sigma V^T$, where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices, and $\Sigma \in \mathbb{R}^{m\times n}$ is a matrix that contains the positive singular values of A.

The minimization problem can then be rewritten using the SVD of A:

$$\min_{x\in\mathbb{R}^n} \|U\Sigma V^T x - b\|^2.$$

By partitioning V and Σ into $V = [V_1|V_2]$ and $\Sigma = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix}$, with $V_1 \in \mathbb{R}^{n\times r}$ and $\Sigma_r \in \mathbb{R}^{r\times r}$ representing a matrix that contains the positive singular values of A.

and similarly partitioning U and b into $U = [U_1|U_2]$ and $b = \begin{bmatrix} b_1 \\ \\ b_2 \end{bmatrix}$, the minimization problem becomes:

$$\min_{x \in \mathbb{R}^n} \|\Sigma_r V_1^T x - U_1^T b\|^2 + \|U_2^T b\|^2.$$

The least-squares solution, x_{ls}, must satisfy:

$$x_{\text{ls}} = V \begin{bmatrix} \Sigma_r^{-1} U_1^T b \\ \\ \tilde{x}_2 \end{bmatrix} = V \begin{bmatrix} \Sigma_r^{-1} \tilde{b}_1 \\ \\ \tilde{x}_2 \end{bmatrix},$$

where $\tilde{x}_2 \in \mathbb{R}^{n-r}$ can be any vector in the $(n-r)$-dimensional subspace, illustrating the non-uniqueness of x_{ls} in the case of an underdetermined system or when A does not have full column rank.Here, $\tilde{b}_1 = U_1^T b$ represents the projection of the vector b onto the column space spanned by the columns of U_1, and V is the orthogonal matrix from the SVD of A.

This formulation highlights that the least-squares solution is directly influenced by the singular values of A and the projection of b onto the column space of U_1. The vector $\tilde{x}_2$ represents the freedom in the choice of solution within the null space of A, emphasizing the multitude of solutions possible when A is rank-deficient.

2.4.5 Problem P2: Minimum-Norm Least-Squares Solution

Problem P2 focuses on achieving the minimum-norm least-squares solution. This objective is critical in scenarios with underdetermined systems, where multiple least-squares solutions exist, and the solution with the smallest norm is preferred for its stability and uniqueness properties.

Theorem 2. *For any initial states $x(0), y(0), z(0)$, and $w(0)$, the distributed algorithm ensures that $z(t)$ converges to the minimum-norm least-squares solution, z^*, such that $z^* \in \mathcal{X}^*$ and $\|z^*\| \leq \|x^*\|$ for all $x^* \in \mathcal{X}^*$.*

Proof for Problem P2: Finding the Minimum-Norm Least-Squares Solution

Given the least-squares problem $\min_{x \in \mathbb{R}^n} \|Ax - b\|^2$ and the SVD of $A = U\Sigma V^T$, the objective is to find the minimum-norm solution z that satisfies $Ax_{\text{ls}} = Az$. We express z in terms of the SVD components as $z = V\tilde{z}$. The goal is to minimize $\|z\|^2$ under the constraint that $Ax_{\text{ls}} = Az$, leading to:

$$\min_{\tilde{z} \in \mathbb{R}^n} \|V\tilde{z}\|^2 = \tilde{z}^T V^T V \tilde{z} = \begin{bmatrix} \tilde{z}_1^T & \tilde{z}_2^T \end{bmatrix} \begin{bmatrix} \tilde{z}_1 \\ \\ \tilde{z}_2 \end{bmatrix} = \|\tilde{z}_1\|^2 + \|\tilde{z}_2\|^2,$$

where $\tilde{z}_1 \in \mathbb{R}^r$ represents the components associated with the non-zero singular values of A, and $\tilde{z}_2 \in \mathbb{R}^{n-r}$ pertains to the null space components. To achieve the minimum-norm, we set $\tilde{z}_2 = \mathbf{0}$, thereby minimizing the expression.

Under the constraint $Ax_{\text{ls}} = Az$, we have:

$$U_1 \Sigma_r V_1^T [V_1 | V_2] \begin{bmatrix} \tilde{x}_1 \\ \\ \tilde{x}_2 \end{bmatrix} = U_1 \Sigma_r V_1^T [V_1 | V_2] \begin{bmatrix} \tilde{z}_1 \\ \\ \tilde{z}_2 \end{bmatrix} = U_1 \Sigma_r \tilde{z}_1.$$

Hence, to minimize the norm and satisfy the constraint, $\tilde{z}_1$ must be $\Sigma_r^{-1} U_1^T b$. Therefore, the minimum-norm solution is:

$$\begin{bmatrix} \tilde{z}_1 \\ \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} \Sigma_r^{-1} U_1^T b \\ \\ 0 \end{bmatrix},$$

and thus,

$$z = [V_1|V_2] \begin{bmatrix} \Sigma_r^{-1} U_1^T b \\ \\ 0 \end{bmatrix} = V_1 \Sigma_r^{-1} U_1^T b.$$

$$z = [V_1|V_2] \begin{bmatrix} \Sigma_r^{-1} \tilde{b}_1 \\ \\ 0 \end{bmatrix} = V_1 \Sigma_r^{-1} \tilde{b}_1,$$

where $\tilde{b}_1 = U_1^T b$ is the projection of b onto the column space of A, represented by the columns of U_1. This approach leverages the structure of A's SVD to efficiently compute the minimum-norm least-squares solution z. This formulation demonstrates that the minimum-norm least-squares solution, z, is achieved by the projection of the vector b onto the column space of A through the inverse of the non-zero singular values Σ_r^{-1}, then transformed by the matrix V_1. This ensures that z not only satisfies the least-squares condition but also that its norm is minimized.

2.4.6 Problem P3: Detecting the Existence of Solutions

Problem P3 is concerned with determining whether a solution to $A\mathbf{x} = \mathbf{b}$ exists, essentially verifying if $\mathbf{b}$ is in the range space of A, $\mathcal{R}(A)$. This verification is pivotal for the nodes to determine the feasibility of finding a valid solution.

Theorem 3. *For any initial states $x(0), y(0), z(0)$, and $w(0)$, the distributed algorithm ensures that $y(t)$ converges 0 if, $\mathbf{b} \in \mathcal{R}(A)$ and to a non-zero vector if $\mathbf{b} \notin \mathcal{R}(A)$.*

The approach to determine whether a solution exists utilizes the property of the system at equilibrium and the behavior of the error vector $\mathbf{y}$ over time. Consider the augmented system equation incorporating the error dynamics:

$$A\mathbf{x}_{\text{ls}} + y = \mathbf{b},$$

where $\mathbf{x}_{\text{ls}}$ represents the least-squares solution. By decomposing A using Singular Value Decomposition (SVD), we express A as $A = U\Sigma V^T$. Partitioning U, Σ, and V and substituting into the equation gives:

$$(U_1\Sigma_r V_1^T)([V_1|V_2]\begin{bmatrix}\tilde{x}_1\\ \\ \tilde{x}_2\end{bmatrix}^T + [U_1|U_2]\begin{bmatrix}\tilde{y}_1\\ \\ \tilde{y}_2\end{bmatrix}^T = [U_1|U_2]\begin{bmatrix}\tilde{b}_1\\ \\ \tilde{b}_2\end{bmatrix}^T.$$

This simplifies to:

$$(U_1\Sigma_r\Sigma_r^{-1}\tilde{b}_1) + U_1\tilde{y}_1 + U_2\tilde{y}_2 = U_1\tilde{b}_1 + U_2\tilde{b}_2.$$

For the system to be in equilibrium with zero error, it requires $\tilde{y}_1 = 0$ and $\tilde{y}_2 = \tilde{b}_2$. Thus, $\mathbf{y} = U_2\tilde{b}_2$.

At equilibrium, considering the system dynamics and the property $A^T\mathbf{y} = 0$ due to orthogonality, we analyze:

$$A^T\mathbf{y} = (V_1\Sigma_r U_1^T)U_2\tilde{b}_2 = 0,$$

indicating that $U_1^T U_2 = 0$ by the properties of orthogonal matrices, ensuring that $\mathbf{y}(\infty) = 0$ aligns with $\mathbf{b}$ being within the range space of A.

This implies that if $\mathbf{y}(\infty) = 0$ across all nodes, then collectively, the network confirms the existence of a solution, as $\mathbf{b}$ lies in the column space of A. Therefore, the distributed computation effectively validates that at least one solution to $A\mathbf{x} = \mathbf{b}$ exists, satisfying the conditions for Theorem 3.

At equilibrium, the system reaches a state where $\dot{y}(t) = 0$, implying that $-Ay(\infty) + b = 0$ or $Ay(\infty) = b$. The condition $y(\infty) = 0$ implies that the system has successfully reduced the residual error to zero, indicating that b is indeed within the range space of A, $R(A)$.

In conclusion, the exploration of Problems P1 to P3 within a distributed setting showcases the profound capability of networked systems to collaboratively address complex computational tasks. Through distributed algorithms, nodes can effectively compute least-squares and minimum-norm solutions and verify the existence of solutions, highlighting the potential for decentralized problem-solving in linear algebraic systems. This investigation not only underscores the practical applications of such distributed methodologies but also sets a foundation for future advancements in networked system optimization and analysis.

2.4.7 Convergence Analysis

This subsection is dedicated to analyzing the convergence properties of the dynamical system governed by the equation (2.23). By deriving the eigenvalues and eigenvectors of the system matrix, we assess the stability and determine the convergence behavior of the state variables $\tilde{x}_i, \tilde{y}_i, \tilde{z}_i$, and $\tilde{w}_i$.

System Dynamics

The dynamical system can be described by the differential equation given in (2.23), where the matrix structure suggests a partitioned approach to understanding the dynamics:

$$\begin{bmatrix} \dot{\tilde{x}}_i \\ \dot{\tilde{y}}_i \\ \dot{\tilde{z}}_i \\ \dot{\tilde{w}}_i \end{bmatrix} = \left[\begin{array}{cc|cc} 0 & \sigma_i & 0 & 0 \\ -\sigma_i & -1 & 0 & 0 \\ \hline 0 & 0 & -1 & \sigma_i \\ \sigma_i & 0 & -\sigma_i & 0 \end{array}\right] \begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ \tilde{z}_i \\ \tilde{w}_i \end{bmatrix} + \begin{bmatrix} 0 \\ \tilde{b} \\ 0 \\ 0 \end{bmatrix}.$$

Eigenvalue Analysis

The stability and convergence of the system heavily depend on the eigenvalues of the matrix governing the dynamics. For the first block, which is representative of the entire matrix dynamics, the eigenvalues are computed as follows:

$$\lambda_1 = \lambda_2 = \left(-1 - \frac{\sqrt{1-4\sigma_i^2}}{2}\right), \quad \lambda_3 = \lambda_4 = \left(-1 + \frac{\sqrt{1-4\sigma_i^2}}{2}\right).$$

The corresponding eigenvectors are:

$$V_1', V_2' = \begin{bmatrix} 0 \\ \frac{-1+\sqrt{1-4\sigma_i^2}}{2\sigma_i} \\ 0 \\ 0 \end{bmatrix}, \quad V_3', V_4' = \begin{bmatrix} 0 \\ \frac{-1-\sqrt{1-4\sigma_i^2}}{2\sigma_i} \\ 0 \\ 0 \end{bmatrix}.$$

Convergence and Stability Discussion

The eigenvalues indicate the rate at which the state variables converge to their steady states. Negative real parts of all eigenvalues suggest that the system is stable and converges to the equilibrium point. The eigenvectors provide insight into the specific modes through which the system approaches stability. Convergence occurs as

$$t \to \infty \text{ if } \Re(\lambda) < 0 \text{ for all } \lambda.$$

The more negative the real part of the eigenvalue, the faster the convergence. Thus, analyzing these eigenvalues allows us to predict the behavior of the system under various initial conditions and input scenarios.

2.5 Algorithm Extension

2.5.1 Problem P4: Computing the Moore-Penrose Inverse

We describe the continuous-time distributed algorithm that enables nodes to cooperatively compute the MPI. Each node updates its state based on local computations and interactions with its neighbors, aiming to converge to the MPI.

Theorem 4. *For any initial states $x(0), y(0), z(0)$, and $w(0)$, the distributed algorithm $z(t)$ ensures convergence to the Moore-Penrose Inverse of matrix A, denoted by $A^\dagger$. This final state $z(\infty)$ at all nodes collectively represents $A^\dagger$ in a distributed matrix form.*

Proof of Convergence

The Moore-Penrose Inverse $A^\dagger$ for a matrix A can be calculated using the Singular Value Decomposition (SVD) of A, given by $A = U\Sigma V^T$. The inverse is expressed as $A^\dagger = V\Sigma^\dagger U^T$, where $\Sigma^\dagger$ is the diagonal matrix consisting of the reciprocals of the non-zero singular values of A, padded with zeros to match the dimensions of A.

The distributed computation of $A^\dagger$ involves each node calculating its corresponding portion of $A^\dagger$ using the local segments of V, Σ, and U that each node maintains:

$$A^\dagger = [V_1|V_2] \begin{bmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix}.$$

$$z(\infty) = [V_1|V_2] \begin{bmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix},$$

where V_1 and U_1 contain the components associated with the non-zero singular values of A, and V_2, U_2 pertain to the null space components.

The initialization of each node with appropriate segments of U, V, and Σ allows for the effective parallel computation of $A^\dagger$. Nodes utilize local data and cooperative interactions with other nodes to iteratively refine their state towards the global objective, converging to the distributed representation of $A^\dagger$.

Observation

The results confirm that the state of each node $z(t)$ converges to a component of $A^\dagger$, validating the effectiveness and robustness of the approach in computing the Moore-Penrose Inverse in a distributed manner.

The successful computation of $A^\dagger$ across the network highlights the practical applicability and efficiency of the distributed approach, showcasing the potential for scalable matrix computations in decentralized network environments.

2.5.2 Problem P5: Detecting Full Column Rank

Problem P5 involves each node in the network determining whether the matrix $A \in \mathbb{R}^{m\times n}$ possesses full column rank, i.e., $\text{rank}(A) = n$. This is crucial as it indicates that the columns of A are linearly independent, which is essential for the uniqueness of solutions in linear systems.

Theorem 5. *Given a network represented as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and assuming arbitrary initialization of the state variables $x(0), y(0), z(0)$, and $w(0)$ as independent and identically distributed Gaussian random variables with zero mean and unit variance, the distributed algorithm can ascertain if A has full column rank by comparing the equilibrium states $x(\infty)$ and $z(\infty)$ of the network nodes.*

Criterion for Detecting Full Column Rank

The nodes employ a continuous-time distributed algorithm with initial conditions as mentioned. The algorithm allows the state variables x, y, z, and w to evolve towards an equilibrium, aiming to analyze the convergence behavior of x and z.

- **Criterion for Full Column Rank:**

  If $x(\infty) = z(\infty)$ for all nodes $i \in \mathcal{V}$, then matrix A has full column rank.

- **Non-Full Column Rank Detection:**

  If $x(\infty) \neq z(\infty)$ for any node $i \in \mathcal{V}$, then matrix A does not have full column rank.

This criterion implies that if the least-squares solution $x(\infty)$ is equivalent to the minimum-norm least-squares solution $z(\infty)$ across the network, then A is uniquely solvable for any arbitrary vector b, indicating that A has linearly independent columns and possesses full column rank.

Outcome Classification:

- **True Positive (TP):** The condition $x(\infty) = z(\infty)$ holds, correctly indicating full column rank.
- **True Negative (TN):** The condition $x(\infty) \neq z(\infty)$ holds, correctly indicating non-full column rank.
- **False Positive (FP):** The scenario where $x(\infty) \neq z(\infty)$ incorrectly suggests full column rank is theoretically impossible due to the rigorous structure of the distributed computation and matrix properties.
- **False Negative (FN):** Although statistically improbable (probability = 0), the condition $x(\infty) = z(\infty)$ might incorrectly suggest non-full column rank under exceptional scenarios.

Implications and Computational Process

This method leverages the distributed computation capabilities of the network to deduce essential structural properties of A, particularly its column rank, through local computations and inter-node communications. The outcome not only illuminates the solvability and characteristics of linear systems associated with A but also underscores the potency of distributed algorithms in matrix analysis and computation.

The analysis of the full column rank condition enhances understanding of the structural integrity and independence of A's columns and serves as a crucial aspect of

distributed matrix operations, reflecting the network's ability to collaboratively solve complex matrix-related problems effectively.

2.5.3 Problem P6: Detecting Full Row Rank

Problem P6 focuses on each node in a network determining whether the matrix $A \in \mathbb{R}^{m \times n}$ has full row rank, i.e., $\text{rank}(A) = m$. This is significant as it indicates that the rows of A are linearly independent, essential for covering the entire row space in linear systems.

Theorem 6. *Given a network modeled as an undirected graph $G = (V, E)$, with the vector b having independent and identically distributed entries with zero mean and unit variance and arbitrary initial conditions $x(0), y(0), z(0)$ and $w(0)$, the distributed algorithm enables nodes to determine if A has full row rank by examining the convergence state of the error vector $y(\infty)$ across the network.*

Criterion for Detecting Full Row Rank

The nodes employ a continuous-time distributed algorithm, allowing their state variables, particularly the error vector y, to evolve towards an equilibrium. The adjustment process is critical to determining the full row rank status of A.

Criterion for Full Row Rank: If $y(\infty) = 0$ for all nodes $i \in V$, then A has full row rank.

Non-Full Row Rank Detection: If $y(\infty) \neq 0$ for any node $i \in V$, then A does not have full row rank.

Outcome Classification

- **True Positive (TP):** The condition $y(\infty) = 0$ holds across all nodes, correctly indicating full row rank.
- **True Negative (TN):** The condition $y(\infty) \neq 0$ holds for any node, correctly indicating non-full row rank.
- **False Positive (FP):** As the detection algorithm is designed based on matrix properties and distributed computation, the occurrence of $y(\infty) \neq 0$ suggesting a full row rank is considered theoretically impossible.
- **False Negative (FN):** While statistically improbable (with probability approaching zero), $fy(\infty) = 0$ might incorrectly suggest non-full row rank under exceptional scenarios.

Implications and Computational Process

This method uses the distributed computational strength of the network to ascertain important structural attributes of A, specifically its row rank, through localized computations and peer-to-peer communication. The results not only shed light on the characteristics and solvability of linear systems linked to A but also emphasize the effectiveness of distributed algorithms in matrix analysis and computations.

2.6 Additional Discussion

2.6.1 LELV Matrix Structure and Initialization

Given the matrix A, characterized by its block structure involving A^T, $-I_m$, and A, the initialization phase accounts for this unique composition, setting the groundwork for targeted local computations and interactions.

1. **Local Data Interpretation:** Each node interprets its portion of A, focusing on the relevant segments $-A$, A^T, and identity matrices. This step includes parsing local information and preparing it for computation.

2. **State Variable Initialization:** Nodes initialize state variables corresponding to their roles in the matrix structure, considering their contribution to either the A or A^T parts of the matrix.

3. **Communication Channels:** Establish communication protocols aligning with the matrix's block structure, ensuring nodes responsible for A and A^T parts can exchange necessary data.

2.6.2 Local Computation Reflecting Matrix Dynamics

The computation phase respects the matrix A's structural dynamics, focusing on operations relevant to the system's interconnected nature.

1. **Block-Specific Operations:** Nodes execute computations based on their segment of A, processing either the A^T, $-A$, or identity matrix components.

2. **Intermediate State Updates:** Calculate intermediate state updates reflecting the influence of matrix dynamics, storing results for communication or further computation.

Chapter 3

Local-Equation Global-Variable Problem

3.1 Introduction

The LEGV paradigm represents a scenario where each node in a distributed system solves a local equation while collectively contributing to the resolution of a global variable. Illustrated in Figure (3.1), this schematic exemplifies the essence of LEGV computations.

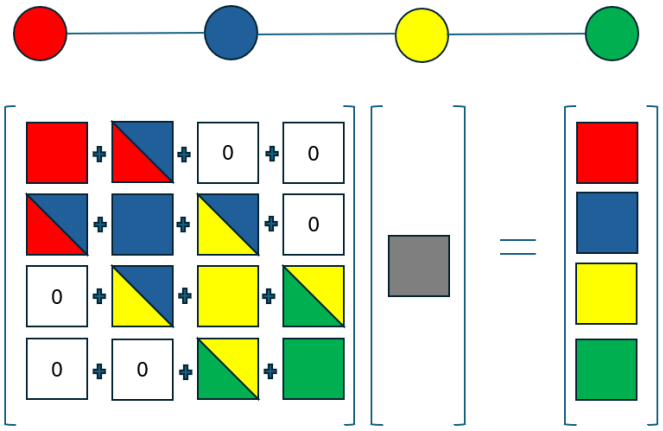

Figure 3.1: The Local-Equation Global-Variable Problem.

In the provided visual, the colored circles epitomize nodes, each of which is charged with solving a local equation. Unlike the LELV model, where each node's equation pertains to entirely local variables, the LEGV framework stipulates that local computations are intimately tied to a global variable.

The matrices, partitioned into blocks colored and marked with diagonals, symbolize the local data each node works with, augmented by the addition operation indicating the synthesis of local computations into a global variable. The zeros mean that segments of the entire dataset of particular nodes have no knowledge or contribution to the global variable, aligning with the defining characteristic of the LEGV approach. The additive contribution of a node's data to the global computation.

The single grey block epresents the aggregation process where all local computations and interactions are combined to form the global solution.

A practical interpretation of this could be as follows: consider the equation $Ax = b$, where A is a matrix, x is the global variable vector, and b is the result vector. In the LEGV context, each node computes a segment of A multiplied by the global x to contribute to the corresponding segment of b. The collective contributions of all nodes' computations yield the full vector b.

3.2 Problem Formulation

Motivation

In many real-world applications, such as sensor networks, distributed control systems, and cooperative robotics, nodes are often required to work collectively to solve complex problems that are inherently distributed. These applications demand efficient algorithms that can handle not only the locality of data but also the global interdependence of variables which influence the overall system dynamics.

Challenges

The primary challenge in LEGV is to design an algorithm that efficiently integrates the influence of global variables while respecting the computational constraints and autonomy of local nodes. This integration must ensure that each node's computations contribute meaningfully to the global system's objectives without necessitating excessive communication or central coordination, which could introduce bottlenecks or vulnerabilities.

Objectives

The objectives of the LEGV problem can be summarized as follows:

- To develop a distributed algorithm that allows each node to process part of the global data while ensuring that local computations align with global system requirements.
- To guarantee that the algorithm converges to a solution that is both accurate and efficient, leveraging the network's topology and the nodes' local data.
- To explore the stability and convergence properties of the proposed algorithm, ensuring that it performs robustly under various network conditions and configurations.

3.3 Algorithm Description

The Local-Equation Global-Variable (LEGV) problem tackles distributed computations in a network graphed as an undirected and connected structure, representing nodes and their interactions. The problem focuses on leveraging both local and global data to achieve systemic solutions across the network.

Matrix Formulation and System Dynamics

Consider a network modeled as an undirected, connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ represents the set of nodes and $\mathcal{E} \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ denotes the edges connecting these nodes. Each node $i \in \mathcal{V}$ has direct communication with its neighbors j, provided $\{i, j\} \in \mathcal{E}$, with the assumption that the communication is error-free and instantaneous.

The nodes are associated with a structured matrix A and a vector b, partitioned as follows:

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ A_{21} & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \cdots & A_{NN} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

where each matrix element A_{ij} is set to zero, $A_{ij} = 0$, if $\{i, j\} \notin \mathcal{E}$, ensuring that elements are only non-zero if nodes i and j are connected by an edge in $\mathcal{E}$. Each b_i is known only to node i, aligning the information distribution with the network structure.

The dimensions of A and b are defined by the network's configuration, where the sum of all individual node dimensions $m_1 + m_2 + \ldots + m_N = m$ and $n_1 = n_2 = \ldots = n_N = n$ accommodate the full matrix $A \in \mathbb{R}^{m \times nN}$ and vector $b \in \mathbb{R}^m$. In which, each $A_i j \in \mathbb{R}^{m_i \times n_j}$ knows $b_i \in \mathbb{R}_i^m$.

This arrangement allows each node to autonomously process its local data and collaboratively solve the global equation $Ax = b$ through coordinated computation and communication within the graph.

The dynamics of the distributed algorithm are governed by the following differential equation, which dictates the evolution of the state variables $\hat{x}(t)$, $x(t)$, $y(t)$, $\hat{z}(t)$, $z(t)$, and $w(t)$ for each node in the network:

$$\begin{bmatrix} \dot{\hat{x}}(t) \\ \dot{x}(t) \\ \dot{y}(t) \\ \dot{\hat{z}}(t) \\ \dot{z}(t) \\ \dot{w}(t) \end{bmatrix} = \begin{bmatrix} 0 & L_n & 0 & 0 & 0 & 0 \\ -L_n & -L_n & A^T & 0 & 0 & 0 \\ 0 & -A & -I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & L_n & 0 \\ 0 & 0 & 0 & -L_n & -I & A^T \\ 0 & A & 0 & 0 & -A & 0 \end{bmatrix} \begin{bmatrix} \hat{x}(t) \\ x(t) \\ y(t) \\ \hat{z}(t) \\ z(t) \\ w(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ b \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where L_n is the Laplacian matrix reflecting the connectivity within the network, A^T represents the transpose of the matrix components associated with each node, and I is the identity matrix corresponding to the dimensions of each state variable block.

3.3.1 Interpretation and Operational Impact

The system dynamics are specifically designed to ensure robust data processing and interaction between nodes. Each node performs computations based on its part of A and b and exchanges information with neighbors to refine its local state estimates. The

global variables are updated based on collective inputs from all nodes, ensuring that the system's overall state evolves toward a solution that satisfies

$$\underbrace{\begin{bmatrix} \sum_{j\in\mathcal{N}_1} A_{1j} \\ \sum_{j\in\mathcal{N}_2} A_{2j} \\ \sum_{j\in\mathcal{N}_3} A_{3j} \\ \vdots \\ \sum_{j\in\mathcal{N}_N} A_{Nj} \end{bmatrix}}_{\mathbb{A}} x = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}}_{\mathbf{b}}$$

3.4 Algorithm Analysis

Instead of performing analysis for an arbitrary graph, in what follows, we condsider a specific graph represented in Figure 3.2. Ths is purely for notation convenience and is without loss of generality because results of the analysis are applicable to a general graph with modification. Consider a network modeled as an undirected, 4-node connected graph. Given the dynamics of the system represented by the state vector derivative $\dot{\tilde{x}}$, the corresponding system matrix incorporating the specified L_n and matrix A is:

Figure 3.2: A 4-node undirected graph used in the LEGV analysis.

$$A = \begin{bmatrix} A_{11} & A_{12} & 0 & 0 \\ A_{21} & A_{22} & A_{23} & 0 \\ 0 & A_{32} & A_{33} & A_{34} \\ 0 & 0 & A_{43} & A_{44} \end{bmatrix}$$

$$x(t) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad z(t) = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix}, \quad w(t) = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

$$y(t) = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}, \quad \hat{x}(t) = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{bmatrix}, \quad \hat{z}(t) = \begin{bmatrix} \hat{z}_1 \\ \hat{z}_2 \\ \hat{z}_3 \\ \hat{z}_4 \end{bmatrix}$$

$L_n = L \otimes I_n$,

$$L_n = \begin{bmatrix} I_n & -I_n & 0 & 0 \\ -I_n & 2I_n & -I_n & 0 \\ 0 & -I_n & 2I_n & -I_n \\ 0 & 0 & -I_n & I_n \end{bmatrix}$$

The system dynamics are expressed as:

$$\begin{bmatrix} \dot{\hat{x}}(t) \\ \dot{x}(t) \\ \dot{y}(t) \\ \dot{\hat{z}}(t) \\ \dot{z}(t) \\ \dot{w}(t) \end{bmatrix} = \begin{bmatrix} 0 & L_n & 0 & 0 & 0 & 0 \\ -L_n & -L_n & A^T & 0 & 0 & 0 \\ 0 & -A & -I_m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & L_n & 0 \\ 0 & 0 & 0 & -L_n & -I_{nN} & A^T \\ 0 & A & 0 & 0 & -A & 0 \end{bmatrix} \begin{bmatrix} \hat{x}(t) \\ x(t) \\ y(t) \\ \hat{z}(t) \\ z(t) \\ w(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ b \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.1}$$

Equilibrium occurs when each derivative is set to zero:

$$\dot{\hat{x}}(t) = 0 = L_n x(t) \tag{3.2}$$

$$\dot{x}(t) = 0 = -L_n \hat{x}(t) - L_n x(t) + A^T y(t) \tag{3.3}$$

$$\dot{y}(t) = 0 = -Ax(t) - y(t) + b \tag{3.4}$$

$$\dot{\hat{z}}(t) = 0 = L_n z(t) \tag{3.5}$$

$$\dot{z}(t) = 0 = -L_n \hat{z}(t) - z(t) + A^T w(t) \tag{3.6}$$

$$\dot{w}(t) = 0 = Ax(t) - Az(t) \tag{3.7}$$

This section details the mathematical foundations and implications of the dynamics described by the differential equations governing the Local-Equation Global-Variable (LEGV) problem, particularly focusing on the properties of the Laplacian matrix and its effects on the system states.

3.4.1 Properties of the Laplacian Matrix

The Laplacian matrix L of any undirected connected graph is symmetric and positive semi-definite (p.s.d.). This implies that all eigenvalues are real and non-negative, with exactly one eigenvalue equal to zero. This zero eigenvalue corresponds to the eigenvector v which is the all-one vector, indicating total consensus among nodes in the absence of external influences.

Fact 1:

$$Lv = \lambda v = 0 \quad \text{where} \quad v = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

In the context of the network system described, the Kronecker product of L with the identity matrix I_n, denoted $L_n = L \otimes I_n$, results in n zero eigenvalues.

Fact 2:

$$L_n \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \implies x_1 = x_2 = \cdots = x_n$$

3.4.2 Implications for System Variables

Given that each x_i converges to a common value $\bar{x}$, as inferred from the properties of L_n, the differential equation for (3.2) becomes:

$$L_n x(t) = 0 \Rightarrow x_1 = x_2 = x_3 = x_4 = \bar{x} \tag{3.8}$$

Consequently, the equation for $\dot{x}(t)$, (3.3) simplifies to:

$$0 = L_n \hat{x}(t) + A^T y(t) \tag{3.9}$$

and the dynamics of $\dot{y}(t)$ (3.4) becomes:

$$\begin{bmatrix} A_{11} + A_{12} \\ A_{21} + A_{22} + A_{23} \\ A_{32} + A_{33} + A_{34} \\ A_{43} + A_{44} \end{bmatrix} \bar{x} + \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \tag{3.10}$$

where

$$\mathbb{A} = \begin{bmatrix} A_{11} + A_{12} \\ A_{21} + A_{22} + A_{23} \\ A_{32} + A_{33} + A_{34} \\ A_{43} + A_{44} \end{bmatrix}$$

Therefore,

$$\mathbb{A}\bar{x} + y = b \tag{3.11}$$

Fact 2 and (3.5) implies that:

$$L_n z(t) = 0 \Rightarrow z_1 = z_2 = z_3 = z_4 = \bar{z} \tag{3.12}$$

Similarly, the implications for $\dot{z}(t)$, (3.6) leads to:

$$L_n \begin{bmatrix} \hat{z}_1 \\ \hat{z}_2 \\ \hat{z}_3 \\ \hat{z}_4 \end{bmatrix} + \begin{bmatrix} \bar{z} \\ \bar{z} \\ \bar{z} \\ \bar{z} \end{bmatrix} = A^T \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \tag{3.13}$$

and ultimately:

$$\mathbb{A}\bar{x} = \mathbb{A}\bar{z} \tag{3.14}$$

demonstrating that $\bar{x}$ and $\bar{z}$ represent the least squares and minimum-norm least-squares solutions respectively. Premultiplying by $[I_n\ I_n\ I_n\ I_n]$ as in equation (3.13).

$$[I_n, I_n, I_n, I_n] \times L_n \begin{bmatrix} \hat{z}_1 \\ \hat{z}_2 \\ \hat{z}_3 \\ \hat{z}_4 \end{bmatrix} + \begin{bmatrix} \bar{z} \\ \bar{z} \\ \bar{z} \\ \bar{z} \end{bmatrix} = [I_n, I_n, I_n, I_n] \times A^T \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \tag{3.15}$$

Define $\mathbb{A}$ and N as follows:

$$\mathbb{A}^T = [I_n, I_n, I_n, I_n] \times A^T,$$

$$N = [I_n, I_n, I_n, I_n] \times L_n = 0$$

This implies that

$$N\bar{z} = \mathbb{A}^T w \tag{3.16}$$

Similarly, if (3.3) is also premultiplied by $[I_n\ I_n\ I_n\ I_n]$, we have:

$$0 = \begin{bmatrix} I_n & I_n & I_n & I_n \end{bmatrix} L_n \hat{x} + \begin{bmatrix} I_n & I_n & I_n & I_n \end{bmatrix} A^T y \tag{3.17}$$

where:

$$\begin{bmatrix} I_n & I_n & I_n & I_n \end{bmatrix} \times L_n = 0 \quad \text{and} \quad \begin{bmatrix} I_n & I_n & I_n & I_n \end{bmatrix} \times A^T = \mathbb{A}^T$$

This implies that

$$\mathbb{A}^T y = 0. \tag{3.18}$$

3.5 Additional Discussion

3.5.1 Comparison of Equilibrium Analysis between LELV and LEGV

The analysis of equilibrium points in both Local-Equation Local-Variable (LELV) and Local-Equation Global-Variable (LEGV) frameworks reveals similarities in the structure of their equations and the implications for system behavior.In both LELV and LEGV, the equilibrium equations involve matrix operations that reflect interactions between the state variables and the system matrix A.

Observation 1

- LELV: $0 = A^T y$ as in equation (2.11) indicates orthogonality between y and the row space of A.
- LEGV: $\mathbb{A}^T y = 0$ as in equation (3.18) indicating that the vector y becomes orthogonal to the row space of the augmented matrix $\mathbb{A}$, which encompasses both the influences of A and L_n, also suggesting orthogonality

Observation 2

- **LELV Equation:**

  Equation (2.13) in the LELV framework indicates a direct transformation where the variable w directly influences the state variable z through the matrix A^T, without the involvement of additional network-specific matrices.

- **LEGV Equation:**

  In the LEGV framework, the equation (3.16) introduces the matrix N, which often incorporates elements like the Laplacian matrix reflecting network topology. This matrix modulates how the aggregated or global variable representation $\bar{z}$ is affected by w through A^T, suggesting a more complex interaction that takes the entire network structure into account.

Observation 3

- **LELV Equation:**

  In LELV, this equation (2.12) suggests a direct interaction between the system matrix A, the state variable x, and the auxiliary variable y to balance the external

input b. The equation emphasizes a straightforward linear relationship without additional complexity from the network topology.

- **LEGV Equation:**

  Similarly, in LEGV, equation (3.11) introduces the augmented matrix $\mathbb{A}$, which typically includes additional network-specific parameters such as the Laplacian matrix. This equation deals with the global state variable $\bar{x}$, reflecting a broader interaction influenced by the network's entire topology alongside the usual contributions from y to match b.

Observation 4

- **LEGV Equation:**

  In the LEGV framework, equation (3.14) suggests that the transformed states $\bar{x}$ and $\bar{z}$ by the augmented matrix $\mathbb{A}$ are equivalent at equilibrium. This equation implies that $\bar{x}$ and $\bar{z}$ are respectively the least squares solution and the minimum norm solution, transformed by the augmented matrix $\mathbb{A}$, which includes network-specific parameters.

- **LELV Equation:**

  Conversely, in LELV, equation (2.14) represents a balance or cancellation of effects between two different transformations of the state variable x under the matrix A. Similarly, x and z in this context represent the least squares solution and the minimum norm solution, respectively, demonstrating that the difference in their transformations through matrix A results in a null effect, indicating equilibrium.

We will resort to numerical simulation to demonstrate the correctness and effectiveness of the proposed algorithm given in equation (3.1).

Chapter 4

Simulation Results

4.1 Local-Equation Local-Variable (LELV)

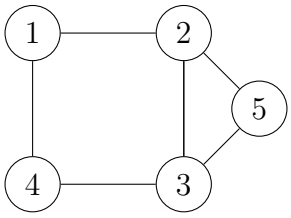

Figure 4.1: A 5-node undirected graph used in the LELV simulations.

The application of the LELV algorithm within a control-theoretic framework on a 5-node undirected graph was explored and represented by a 7×8 matrix A and a vector b. This approach aimed to simulate the minimum-norm least-squares solution (P2) across the network. The results illustrated the algorithm's capability to efficiently distribute the computation load across the network nodes, significantly enhancing the system's scalability and performance.

4.2 Simulation Setup

Matrix and Vector Initialization

The LELV (Local-Equation Local-Variable) problem was simulated using MATLAB to model the dynamics and interactions within a distributed system. Key simulation parameters are described as follows:

- **Matrix** A: A matrix $A \in \mathbb{R}^{m \times n}$ was generated to explore various computational scenarios. The matrix dimensions and the rank were set to:
  - $m = 7$ (number of rows)
  - $m_1 = 1, m_2 = 2, m_3 = 1, m_4 = 2, m_5 = 1$
  - $n = 8$ (number of columns)
  - $n_1 = 2, n_2 = 1, n_3 = 2, n_4 = 1, n_5 = 2$
  - Rank of $A = 4$

The matrix A is given by:

$$A = \left[\begin{array}{cccccccc} 1 & 2 & 3 & 0 & 0 & 1 & 0 & 0 \\ \hline 6 & 8 & 10 & 12 & 2 & 2 & 0 & 2 \\ \hline 3 & 4 & 5 & 6 & 1 & 1 & 0 & 1 \\ \hline 0 & 0 & 4 & 5 & 6 & 7 & 0 & 3 \\ \hline 8 & 2 & 0 & 2 & 16 & 4 & 0 & 0 \\ \hline 4 & 1 & 0 & 1 & 8 & 2 & 0 & 0 \\ \hline 0 & 0 & 8 & 10 & 12 & 14 & 0 & 6 \end{array}\right]$$

Matrix A was structured to include specific properties such as sparsity or particular patterns that are relevant to the LELV problem. The rank condition was chosen to test the algorithm's effectiveness in scenarios of underdetermined systems, where $\text{rank}(A) < \min(m, n)$.

- **Vector** b:The vector b is a 7×1 random vector where each element is drawn from a standard normal distribution, representing external inputs to the system.

- The output matrix C is set as an identity matrix of size $2m + 2n$, facilitating the observation of all state variables directly.

- The matrix D is a zero matrix matching the dimensions of C, indicating no direct feedthrough from the input to the output, which is a common setup in state-space representations where the system dynamics are emphasized.

4.2.1 Initial Conditions for State Variables

State variables x, y, z, and w were initialized to study their evolution over the simulation. The initialization aimed to examine:

- The impact of different starting values on x and w, which are sensitive to initial conditions. Various initial states were tested to observe potential convergence issues or dependencies.
- The behavior of y and z, which were expected to show convergence to a stable solution regardless of their initial values, demonstrating the algorithm's robustness.

4.3 Simulation Results and Discussion

4.3.1 Results

Simulation outcomes were detailed through graphical representations showing the evolution of x, y, z, and w. These plots provide insights into:

- **Dependency on Initialization**: Observations on how x and w reacted to different initial conditions, highlighting the need for careful selection of initial states in practical deployments.
- **Stability of y and z**: Despite variations in initialization, both y and z converged reliably, indicating the algorithm's effectiveness in achieving consistent results.

4.3.2 Discussion

The behavior of the state variables underlines the necessity for a well-considered setup in LELV problems, especially in complex network scenarios. The findings from this

simulation guide the implementation of LELV algorithms in real-world applications, emphasizing the importance of initial conditions and robust algorithmic design.

4.4 Conclusion and Insights

The simulation results for the Local-Equation Local-Variable (LELV) system over a period of $T = 0 : 0.001 : 10$ seconds provided critical insights into the operational dynamics and convergence behaviors under various conditions. The algorithm demonstrated robustness and adaptability, showcasing its potential to handle network-based challenges effectively.

As illustrated in Figures 4.2, 4.3, 4.4, 4.5, the dynamic responses of the state variables exhibited notable trends:

- **State Variable** $x(t)$:
  - The plot indicates an initial transient response with some state variables exhibiting oscillatory behavior which settles over time.
  - After the initial period, the state variables appear to reach a steady state, with $x_4(t)$ to $x_8(t)$ demonstrating relatively constant values.
- **State Variable** $y(t)$:
  - Similar to $x(t)$, there is an initial transient period where all state variables converge to steady state values relatively quickly.
  - All state variables of $y(t)$ stabilize near zero, indicating a reduction of the error over time.
- **State Variable** $z(t)$:

- This plot also shows a transient response initially, with all states converging to steady states.
- The variables $z_1(t)$ to $z_8(t)$ display a damped response, suggesting that the system is stable and the least square solution is effectively minimizing the norm.

- **State Variable** $w(t)$:
  - The initial response is quite pronounced with higher amplitude oscillations compared to $x(t)$, $y(t)$, and $z(t)$.
  - The oscillations damp out, and the variables settle down to steady states, indicating the system's stabilization over time.
  - The trends imply that the Lagrange multipliers are adjusting as the optimization progresses towards a solution that satisfies all constraints.

The absence of sustained oscillatory behavior or divergence after the initial transient across all plots suggests that the system represented by the state variables is stable. The convergence of the state variables towards constant values indicates that any perturbations are being effectively managed by the system's inherent properties or control mechanisms.

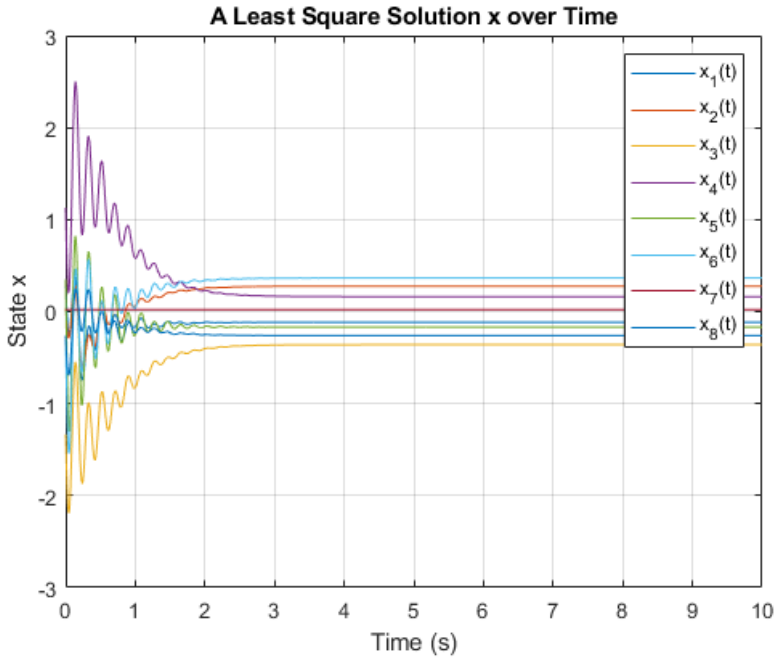

Figure 4.2: Time evolution of state variables $x(t)$

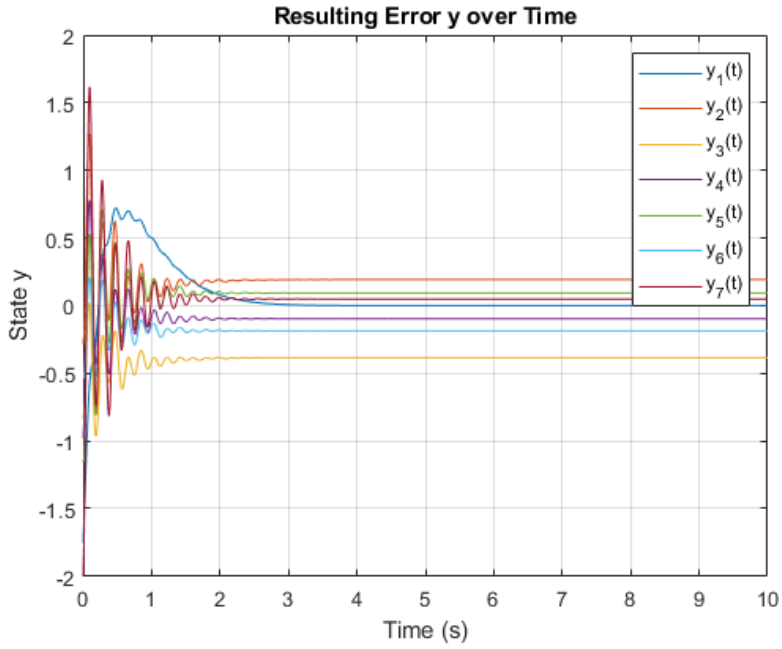

Figure 4.3: Time evolution of state variables $y(t)$

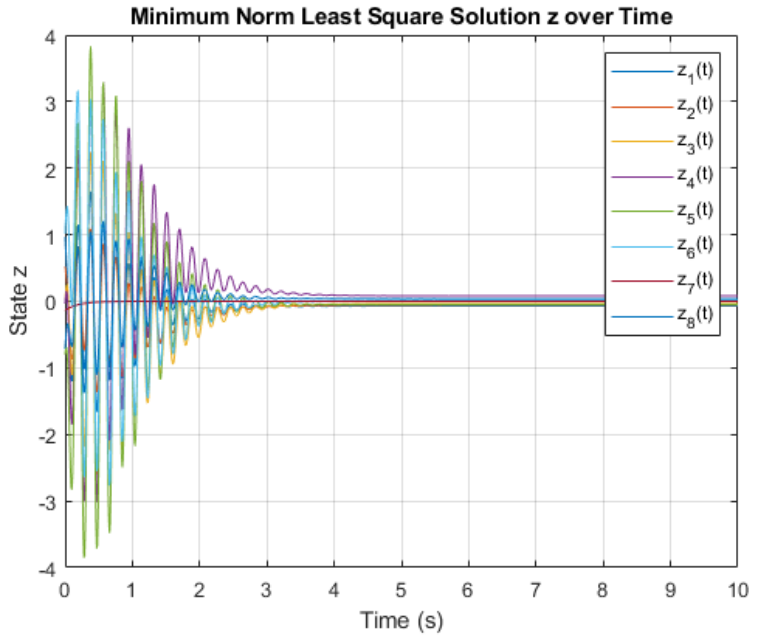

Figure 4.4: Time evolution of state variables $z(t)$

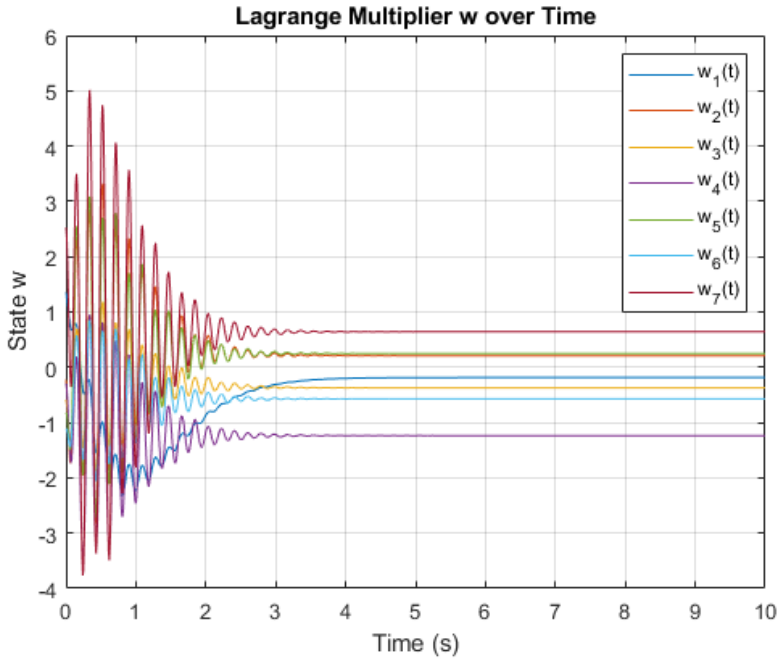

Figure 4.5: Time evolution of state variables $w(t)$

4.5 Local-Equation Global-Variable (LEGV)

Similarly, the LEGV algorithm was analyzed under different network scenarios to assess its capability in global variable computation. The results confirmed that LEGV effectively utilizes local computations to influence global outcomes, optimizing the use of resources and processing time. The matrices and vectors for the simulation of the Local-Equation Global-Variable (LEGV) problem are initialized as follows:

Figure 4.6: A 3-node undirected graph used in the LEGV simulations.

- **Dimension Definitions:**
  - $m = 3$: Number of outputs or measurements.
  - $n = 1$: Number of inputs or control variables per node.
  - $N = 3$: Number of nodes in the network.
  - $nN = n \times N$: Total state dimension size, affected by both n and N.
- **System Matrices:**
  - Matrix $A \in \mathbb{R}^{m \times nN}$ is defined to encapsulate the system dynamics. In this simulation setup, A is given as:

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 4 & 5 \\ 0 & 6 & 7 \end{bmatrix}.$$

– The Laplacian matrix L_n represents the connectivity of the nodes within the network. For a graph with N nodes, L_n is defined as:

$$L_n = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}.$$

– Vector $b \in \mathbb{R}^m$ is initialized randomly to simulate various scenarios, using a standard normal distribution:

$$b = \text{randn}(m, 1).$$

The matrices are structured to ensure that each node has access to local information and can communicate with its neighbors according to the network's topology, while vector b provides the initial conditions for the simulation.

The simulation results for the Local-Equation Global-Variable (LEGV) system are indicative of the system's ability to reach and maintain a stable equilibrium. As shown in Figure 4.7, the dynamic response and convergence behavior of the system were captured over a simulation period, providing insights into the algorithm's robustness under various network conditions.

- **State Variable $\hat{x}(t)$:** The plot indicates an initial transient response with some state variables exhibiting oscillatory behavior which settles over time. After the initial period, the state variables appear to reach a steady state, demonstrating relatively constant values.

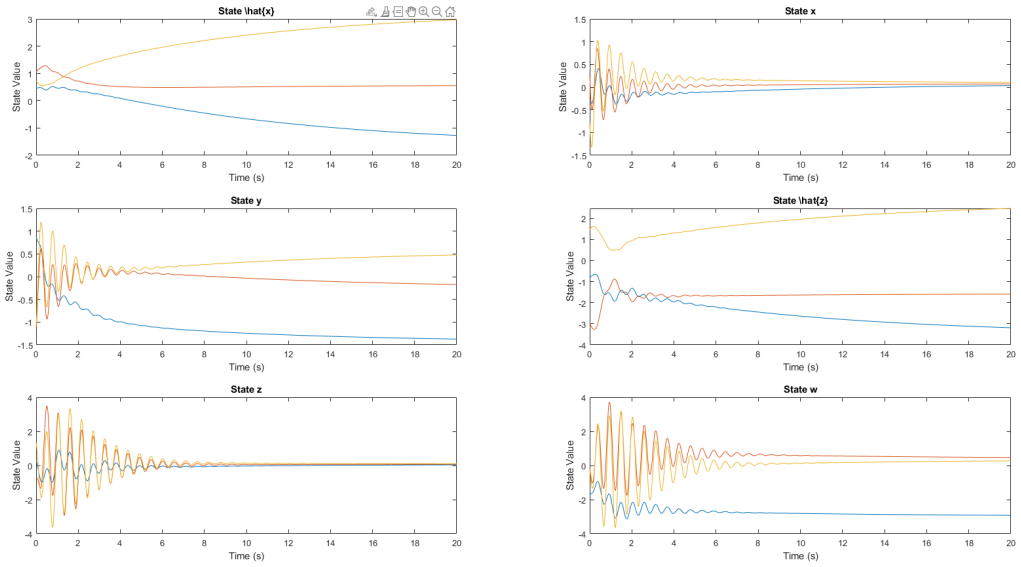

Figure 4.7: Simulation results for the LEGV algorithm.

- **State Variable** $x(t)$**:** Similar to $\hat{x}(t)$, there is an initial transient period where all state variables converge to steady state values relatively quickly. The variables stabilize near zero, indicating a reduction of the error over time.

- **State Variable** $y(t)$**:** The plot also shows a transient response initially, with all states converging to steady states. The variables display a damped response, suggesting that the system is stable.

- **State Variable** $\hat{z}(t)$**:** The initial response is quite pronounced with higher amplitude oscillations compared to other state variables. The oscillations damp out, and the variables settle down to steady states, indicating the system's stabilization over time.

- **State Variable** $z(t)$**:** Exhibits behavior similar to $\hat{z}(t)$, but with a slightly different transient response. The convergence to steady state suggests that the least square solution is effectively minimizing the norm.

- **State Variable $w(t)$:** Shows the most complex behavior with higher amplitude oscillations. The trend towards stabilization is indicative of the system's capability to handle perturbations and maintain equilibrium.

The absence of sustained oscillatory behavior or divergence after the initial transient across all plot suggests that the system represented by the state variables is stable. The convergence of the state variables towards constant values indicates that any perturbations are being effectively managed by the system's inherent properties or control mechanisms.

4.6 State Variable Dependencies

The state variables for the LEGV problem are denoted by $\hat{x}(t)$, $x(t)$, $y(t)$, $\hat{z}(t)$, $z(t)$, and $w(t)$. Their behavior with respect to initial conditions can be summarized as follows:

- **State Variables Dependent on Initialization:**
  - $\hat{x}(t)$ and $x(t)$: These state variables represent the system states and are directly affected by the initial conditions. Any change in the initialization of $\hat{x}(t)$ and $x(t)$ will alter their trajectories over time.
  - $w(t)$: As a representation of Lagrange multipliers or dual variables, $w(t)$ is also sensitive to initial conditions, impacting the optimization dynamics of the system.
- **State Variables Independent of Initialization:**
  - $y(t)$ and $z(t)$: These variables tend to converge to a steady state that is determined by the system dynamics and the graph topology, rather than by their initial values.

– $\hat{z}(t)$: Similar to $z(t)$, the state $\hat{z}(t)$ eventually reaches an equilibrium that is independent of its initial conditions, as it is designed to estimate a steady-state that complies with the system's constraints.

Chapter 5

Summary and Conclusions

This thesis presents a novel continuous-time distributed algorithm for complex matrix computations on networked systems. It addresses limitations in prior work concerning distributed data and computations. The algorithm demonstrates robustness and adaptability for diverse matrix operations, including Moore-Penrose inverse computation, matrix rank determination, and solving linear equations in distributed environments. Global exponential convergence is established through theoretical analysis and simulations.

A key differentiation lies in the comprehensive analysis of Local-Equation Local-Variable (LELV) problems. Local-Equation Global-Variable (LEGV) problems were less extensively explored, presenting a promising avenue for future research, particularly regarding algorithm performance optimization and network-wide computational strategies.

Future work could also focus on developing robust termination criteria for the algorithm, enhancing its practical applicability and efficiency.

In conclusion, this thesis contributes significantly to distributed matrix analysis research and lays a foundation for advancements in distributed systems. It addresses critical knowledge gaps and offers practical solutions for networked computational environments, paving the way for more sophisticated and scalable computational methods.

Chapter 6

Appendix

```
% MATLAB code for LELV simulation
close all;
% New system dimensions reflecting the 7x8 matrix A
m = 7; % New definition based on A's row count
n = 8; % New definition based on A's column count
% New system matrix and vector reflecting the 5-node graph scenario
A = [1 2 3 0 0 1 0 0;
     6 8 10 12 2 2 0 2;
     3 4 5 6 1 1 0 1;
     0 0 4 5 6 7 0 3;
     8 2 0 2 16 4 0 0;
     4 1 0 1 8 2 0 0;
     0 0 8 10 12 14 0 6];
b = randn(m,1); % Random vector b with m elements
% Display the rank of matrix A
disp(['Rank of matrix A: ', num2str(rank(A))]);
% Moore-Penrose pseudoinverse solution
xpinv = pinv(A)*b;
% Since A is now 7x8, we don't compute xls as before because A'*A is not
    invertible
% Simulation time
T = 0:0.01:10;
% Adjust initial state size based on new system structure
```

```
% Here, X0 size is hypothetical, assuming a system that can incorporate
    the 7x8 A matrix
X0 = randn(2*m+2*n,1); % Adjusted size for initial state vector
% Adjusted Input signal
u = ones(size(T));
% Hypothetical state-space representation placeholder
AA = [zeros(n) A' zeros(n) zeros(n,m);
-A -5*eye(m) zeros(m,n) zeros(m);
zeros(n) zeros(n,m) -5*eye(n) A';
A zeros(m) -A zeros(m)];
B = [zeros(n,1); b; zeros(n,1); zeros(m,1)];
C = eye(2*m+2*n);
D = zeros(2*m+2*n,1);
% Create system object
sys = ss(AA, B, C, D);
% Simulate system response
X = lsim(sys, u, T, X0)';
% Extract state variables
x = X(1:n,:);
y = X(n+1:n+m,:);
z = X(n+m+1:n+m+n,:);
w = X(n+m+n+1:end,:);
% Extract final values
final_x = x(:, end);
final_y = y(:, end);
final_z = z(:, end);
final_w = w(:, end);
% Print final values
disp('Final values of x:');
disp(final_x);
disp('Final values of y:');
```

```
disp(final_y);
disp('Final values of z:');
disp(final_z);
disp('Final values of w:');
disp(final_w);
% Plotting
figure('Color', 'w');
colors = 'rgbcmyk';
figure('Name', 'State Variables Over Time');
% Plot for x
figure('Name', 'State Variable x over Time', 'Color', 'w');
plot(T, x);
title('A Least Square Solution x over Time');
xlabel('Time (s)');
ylabel('State x');
legend(arrayfun(@(i) sprintf('x_%d(t)', i), 1:size(x,1), 'UniformOutput',
    false));
grid on;

% Plot for y
figure('Name', 'State Variable y over Time', 'Color', 'w');
plot(T, y);
title('Resulting Error y over Time');
xlabel('Time (s)');
ylabel('State y');
legend(arrayfun(@(i) sprintf('y_%d(t)', i), 1:size(y,1), 'UniformOutput',
    false));
grid on;

% Plot for z
figure('Name', 'State Variable z over Time', 'Color', 'w');
```

```
plot(T, z);
title('Minimum Norm Least Square Solution z over Time');
xlabel('Time (s)');
ylabel('State z');
legend(arrayfun(@(i) sprintf('z_%d(t)', i), 1:size(z,1), 'UniformOutput',
    false));
grid on;

% Plot for w
figure('Name', 'State Variable w over Time', 'Color', 'w');
plot(T, w);
title('Lagrange Multiplier w over Time');
xlabel('Time (s)');
ylabel('State w');
legend(arrayfun(@(i) sprintf('w_%d(t)', i), 1:size(w,1), 'UniformOutput',
    false));
grid on;
```

Listing 6.1: LELV Algorithm Simulation Code

```
% MATLAB code for LEGV simulation
clear all;
close all;

% Define the dimensions and matrices
m = 3;
n = 1;
N = 3;
nN = n * N; % Total dimension size affected by n and N

% Example matrices
A = [1 2 0; 3 4 5; 0 6 7];
```

```
b = randn(m,1);

% Laplacian matrix L_n for a graph with N nodes (as an example)
L_n = [1 -1 0; -1 2 -1; 0 -1 1];

% Redefine AA with the new structure
AA = blkdiag(zeros(nN), -L_n, zeros(m), zeros(nN), -eye(nN), zeros(nN,m))
    + ...
    [zeros(nN), L_n, zeros(nN, m), zeros(nN), zeros(nN), zeros(nN, m);
     -L_n, -L_n, A', zeros(nN, m), zeros(nN), zeros(nN, m);
     zeros(m, nN), -A, -eye(m), zeros(m, nN), zeros(m, nN), zeros(m);
     zeros(nN), zeros(nN), zeros(nN, m), zeros(nN), L_n, zeros(nN, m);
     zeros(nN), zeros(nN), zeros(nN, m), -L_n, -eye(nN), A';
     zeros(nN, m), A, zeros(nN, m), zeros(nN, m), -A, zeros(nN, m)];

B = [zeros(nN, 1); zeros(nN, 1); b; zeros(nN, 1); zeros(nN, 1); zeros(m,
    1)];

C = eye(size(AA, 1));
D = zeros(size(B));

% Simulation setup
sys = ss(AA, B, C, D);
T = 0:0.01:20;
X0 = randn(size(AA, 1), 1);
u = ones(size(T));

% Simulate the system
X = lsim(sys, u, T, X0)';

% Extract and plot the relevant states
```

```
hat_x = X(1:Nn, :);
x = X( nN+1:2*nN, :);
y = X(2*nN+1:2*nN+m, :);
hat_z = X( 2*nN+m+1:3*nN+m, :);
z = X( 3*nN+m+1:4*nN+m, :);
w = X(4*nN+m+1:end, :);
% Plotting
figure;
subplot(3, 2, 1);
plot(T, hat_x);
title('State \hat{x}');
xlabel('Time (s)');
ylabel('State Value');

subplot(3, 2, 2);
plot(T, x);
title('State x');
xlabel('Time (s)');
ylabel('State Value');

subplot(3, 2, 3);
plot(T, y);
title('State y');
xlabel('Time (s)');
ylabel('State Value');

subplot(3, 2, 4);
plot(T, hat_z);
title('State \hat{z}');
xlabel('Time (s)');
ylabel('State Value');
```

```
subplot(3, 2, 5);
plot(T, z);
title('State z');
xlabel('Time (s)');
ylabel('State Value');

subplot(3, 2, 6);
plot(T, w);
title('State w');
xlabel('Time (s)');
ylabel('State Value');
```

Listing 6.2: LEGV Algorithm Simulation Code

References

[1] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[2] J. Lu, C. Y. Tang, P. R. Regier, and T. D. Bow, "Gossip algorithms for convex consensus optimization over networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 12, pp. 2917–2923, 2011.

[3] C. Asensio-Marco and B. Beferull-Lozano, "Accelerating consensus gossip algorithms: Sparsifying networks can be good for you," in *2010 IEEE International Conference on Communications*, pp. 1–5, 2010.

[4] X. Yi, X. Li, L. Xie, and K. H. Johansson, "Distributed online convex optimization with time-varying coupled inequality constraints," *IEEE Transactions on Signal Processing*, vol. 68, pp. 731–746, 2020.

[5] J. Lu and C. Y. Tang, "Zero-gradient-sum algorithms for distributed convex optimization: The continuous-time case," *IEEE Transactions on Automatic Control*, vol. 57, no. 9, pp. 2348–2354, 2012.

[6] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

[7] S.-H. Son, M. Chiang, S. Kulkarni, and S. Schwartz, "The value of clustering in distributed estimation for sensor networks," in *2005 International Conference on Wireless Networks, Communications and Mobile Computing*, vol. 2, pp. 969–974 vol.2, 2005.

[8] P. Wang, S. Mou, J. Lian, and W. Ren, "Solving a system of linear equations: From centralized to distributed algorithms," *Annual Reviews in Control*, vol. 47, pp. 306–322, 2019.

[9] X. Wang, S. Mou, and B. D. O. Anderson, "Scalable, distributed algorithms for solving linear equations via double-layered networks," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 1132–1143, 2020.

[10] M. Yang and C. Y. Tang, "A distributed algorithm for solving general linear equations over networks," in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 3580–3585, 2015.

[11] S. Mou and A. S. Morse, "A fixed-neighbor, distributed algorithm for solving a linear algebraic equation," in *2013 European Control Conference (ECC)*, pp. 2269–2273, 2013.

[12] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[13] H. Pan, X. Yu, and G. Wen, "Robust consensus of fractional-order singular uncertain multi-agent system under undirected graph," in *2018 IEEE 27th International Symposium on Industrial Electronics (ISIE)*, pp. 1161–1166, 2018.

[14] H. Moradian and S. S. Kia, "Cluster-based distributed augmented lagrangian algorithm for a class of constrained convex optimization problems," *Automatica*, vol. 129, p. 109608, 2021.

[15] K. I. Tsianos and M. G. Rabbat, "Distributed strongly convex optimization," in *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 593–600, 2012.

[16] Z. Deng and J. Luo, "Fully distributed algorithms for constrained nonsmooth optimization problems of general linear multiagent systems and their application," *IEEE Transactions on Automatic Control*, vol. 69, no. 2, pp. 1377–1384, 2024.

[17] R. Saber and R. Murray, "Agreement problems in networks with directed graphs and switching topology," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, vol. 4, pp. 4126–4132 vol.4, 2003.

[18] M. Zhu and S. Martinez, "Dynamic average consensus on synchronous communication networks," in *2008 American Control Conference*, pp. 4382–4387, 2008.

[19] G. Battistelli, L. Chisci, G. Mugnai, A. Farina, and A. Graziano, "Consensus-based linear and nonlinear filtering," *IEEE Transactions on Automatic Control*, vol. 60, no. 5, pp. 1410–1415, 2015.

[20] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pp. 63–70, 2005.

[21] L. Moreau, "Leaderless coordination via bidirectional and unidirectional time-dependent communication," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, vol. 3, pp. 3070–3075 Vol.3, 2003.

[22] J. Lu and C. Y. Tang, "A distributed algorithm for solving positive definite linear equations over networks with membership dynamics," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 215–227, 2018.

[23] L. Moreau, "Stability of continuous-time distributed consensus algorithms," in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, vol. 4, pp. 3998–4003 Vol.4, 2004.

[24] S. Mou, J. Liu, and A. S. Morse, "A distributed algorithm for solving a linear algebraic equation," *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 2863–2878, 2015.

[25] Z. Li, X. Liu, P. Lin, and W. Ren, "Consensus of linear multi-agent systems with reduced-order observer-based protocols," *Systems and Control Letters*, vol. 60, no. 7, pp. 510–516, 2011.

[26] D. Kingston and R. Beard, "Discrete-time average-consensus under switching network topologies," in *2006 American Control Conference*, pp. 6 pp.–, 2006.

[27] H. Moradian and S. S. Kia, "A study on accelerating average consensus algorithms using delayed feedback," *IEEE Transactions on Control of Network Systems*, vol. 10, no. 1, pp. 157–168, 2023.

[28] X. Zeng, J. Lei, and J. Chen, "Dynamical primal-dual nesterov accelerated method and its application to network optimization," *IEEE Transactions on Automatic Control*, vol. 68, no. 3, pp. 1760–1767, 2023.

[29] S. S. Kia, J. Wei, and L. Chen, "Distributed optimal resource allocation using transformed primal-dual method," in *2023 American Control Conference (ACC)*, pp. 198–203, 2023.

[30] C. Asensio-Marco and B. Beferull-Lozano, "Energy efficient consensus over directed graphs," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4124–4128, 2018.

[31] X. Wang, S. Mou, and D. Sun, "Improvement of a distributed algorithm for solving linear equations," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 4, pp. 3113–3117, 2017.

[32] W. Chen and W. Ren, "Event-triggered zero-gradient-sum distributed consensus optimization over directed networks," *Automatica*, vol. 65, pp. 90–97, 2016.

[33] R. Saha, S. Rini, M. Rao, and A. Goldsmith, "Decentralized optimization over noisy, rate-constrained networks: How we agree by talking about how we disagree," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5055–5059, 2021.

[34] E. Montijano, G. Oliva, and A. Gasparri, "Distributed estimation and control of node centrality in undirected asymmetric networks," *IEEE Transactions on Automatic Control*, vol. 66, no. 5, pp. 2304–2311, 2021.

[35] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[36] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation and consensus using linear iterative strategies," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 650–660, 2008.

[37] T. Charalambous and C. N. Hadjicostis, "Laplacian-based matrix design for finite-time average consensus in digraphs," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 3654–3659, 2018.

[38] N. D. Sayin, Muhammed O.and Vanli, S. S. Kozat, and B. Tamer, "Stochastic subgradient algorithms for strongly convex optimization over distributed networks," *IEEE Transactions on Network Science and Engineering*, vol. 4, no. 4, pp. 248 – 260, 2017.

[39] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[40] N. E. Manitara and C. N. Hadjicostis, "Distributed stopping for average consensus in digraphs," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 957–967, 2018.