

PH.D. IN COMPLEX SYSTEMS FOR PHYSICAL,
SOCIO-ECONOMIC AND LIFE SCIENCES
UNIVERSITÀ DEGLI STUDI DI CATANIA



Università
di Catania

Department of Physics and Astronomy
PhD program in Complex Systems – 38° cycle

MASFENON: A MULTI-AGENT HPC
FRAMEWORK FOR DYNAMIC SIMULATIONS OF
INTERCONNECTED COMPLEX SYSTEMS

Giorgio Locicero
ID number 1000024196

Supervisor: Prof. **Alfredo Pulvirenti**

Coordinator: Prof. **Andrea Rapisarda**

Academic year
2024/2025

Acknowledgements

I would like to express my deepest gratitude to all the people who have supported me throughout this journey and helped me overcome difficult times. Without their encouragement, I would not have been able to complete this work and, above all, this Ph.D.

I am especially grateful to my supervisor for their guidance, for supporting my research, and for always being available to discuss uncertainties and challenges. I also wish to thank my collaborators and tutors, both in Catania and at Teesside University, for the stimulating discussions, brainstorming sessions, and joint research activities, which were not only insightful but also kept my mind constantly engaged.

Finally, I would like to thank my mother for her unwavering mental and emotional support, which has been invaluable to me.

List of Publications

- MASFENON: Multi-Agent Adaptive Simulation Framework for Evolution in Networks of Networks [51] (Published)
- MASFENON: implementing a multi-agent simulation framework for interconnected networks with distributed programming [50] (Published)
- Measuring cancer driving force of chromosomal aberrations through multi-layer Boolean implication networks [21] (Published)
- Bridging Modalities with AI: A Comprehensive Review of AI Innovations in Multimodal Biomedical Imaging (Under Editorial assessment in Nature)
- MASFENON, a novel simulation framework for dynamic interconnected networks: a case study on epidemics networks [49] (Submitted to the Information Sciences journal, Pre-print available)

Contents

Acknowledgements	ii
List of Publications	iv
Introduction	1
1 Background and related work	6
1.1 Overview of sequencing techniques	7
1.1.1 Bulk sequencing	8
1.1.2 Single cell sequencing	11
1.1.3 Comparison between Bulk and Single-Cell RNA Se- quencing	17
1.2 Overview of cell-to-cell communication	19
1.2.1 Cell-Cell Interaction (CCI)	20
1.2.2 Cell-Cell Communication (CCC)	20
1.2.3 Mechanisms of Cell-Cell Communication	21
1.2.4 Computational Inference of Cell-Cell Communication .	21
1.2.5 Applications in Health and Disease	29
1.2.6 Challenges and Future Directions	29
1.2.7 Conclusion	30
1.3 Overview of networks	30
1.3.1 Introduction to Networks Across Domains	30
1.3.2 Fundamental Network Properties	31
1.3.3 Types and Representations of Networks	34
1.3.4 Modeling Real-World Networks	36
1.3.5 Networks Dynamics and Flow Processes	39
1.3.6 From Data to Network: Construction Challenges	42
1.3.7 Biological Networks in Focus	43
1.3.8 Why General Network Simulation Frameworks Are Needed	48
1.4 Overview of Multi-Agent Systems	49
1.4.1 Introduction to Multi-Agent Systems (MAS)	49
1.4.2 Core Characteristics of MAS	52
1.4.3 Agent Architectures	54

1.4.4	Communication and Coordination	55
1.4.5	Organizational Structures in MAS	57
1.4.6	Applications of MAS	59
1.4.7	Challenges and Research Directions	60
1.4.8	MAS Platforms and Frameworks	63
1.5	Synthesis of the Relevant Overviews	65
2	State of the Art Models	66
2.1	Review of Existing Simulation Frameworks	67
2.2	Evaluation of State-of-the-Art Models	69
2.3	Comparison and Innovation: MASFENON's Contributions . .	71
3	The MASFENON Framework	75
3.1	Concept and Design	81
3.1.1	Defining Multi-Agent Interactions in Interconnected Networks	82
3.1.2	Graph Augmentation and Structural Modeling	84
3.1.3	Temporal Network Structure	88
3.1.4	UML Diagram of the Classes	89
3.1.5	Dynamical Principles and Core Modeling Functions . .	91
3.2	Implementation Details	102
3.2.1	Customization of the framework	103
3.2.2	Use of OpenMPI for Inter-Agent Communication . . .	106
3.2.3	Armadillo and OpenMP for Intra-Propagation	107
3.2.4	Scalability Considerations and Optimization Techniques	108
3.2.5	Programming Practices and Development Workflow . .	108
3.2.6	Control Over Memory Leaks	111
3.3	Computational Complexity and Performance Analysis	112
3.3.1	Time Complexity	112
3.3.2	Space Complexity	113
3.3.3	Parallelization Techniques and Efficiency	114
3.4	Additional Functionalities of MASFENON	115
3.4.1	Graph Statistics and Structural Metrics	116
3.4.2	Visualization and Output Analysis Tools	116
3.4.3	Parameter Fitting and Integration with Real Data . .	117
4	Experimental Results	119
4.1	Performance Benchmarking	119
4.1.1	Comparison with Other Frameworks	119
4.1.2	Scaling Behavior in HPC Environments	123
4.2	Epidemiological Simulations	125
4.2.1	Modeling Rationale and Graphical Overview	125
4.2.2	Simulation Setup and Methodology	126
4.2.3	Formal Validation: Statistical and Behavioral Analysis	126

4.2.4	Comparison with Compartmental Models	129
4.2.5	Parameter Sensitivity and Model Fitting	130
4.2.6	Scalability and Performance	132
4.2.7	Conclusions and Future Directions	133
4.3	Biological Simulations	133
4.3.1	Introduction and Rationale	133
4.3.2	Data Sources and Preprocessing	136
4.3.3	Multi-Layer Network Architecture	139
4.3.4	Simulation Setup and Parameter Fitting	149
4.3.5	Discussion and Limitations	155
5	Future Work and Extensions	159
5.1	Enhancements for General Use and Flexibility	159
5.2	Enhancements Specific to Biological Modeling	160
5.3	Integration with Real-World Biological Data	160
5.4	Potential Optimizations for Exascale Computing	161
5.5	User Interaction and Visualization	161
6	Use Cases and Applications	162
6.1	Simulating Cell-To-Cell communication	162
6.2	Epidemic Spread Modeling	163
6.3	Social Network Diffusion	164
6.3.1	Motivation and Context	164
6.3.2	Model Design and Simulation Setup	164
6.3.3	Data and Implementation	165
6.3.4	Simulation Insights	165
6.3.5	Discussion and Use Case Potential	165
6.4	Economics and Organizational Networks	166
	Conclusions	167
	Future Directions	167
	Contributions and Acknowledgments	168
	Writing Process and Use of Large Language Models	169
	Final Remarks	169
A	Appendix: Mathematical Foundation	171
A.1	Linear algebra	171
A.1.1	Vectors and Vector Spaces	171
A.1.2	Linear Independence, Basis, and Dimension	173
A.1.3	Change of Basis	174
A.1.4	Inner Products, Norms, and Orthogonality	176
A.1.5	Matrices and Operations	177
A.1.6	Eigenvalues and Eigenvectors	179
A.1.7	Matrix Decompositions and Inverse	181

A.1.8 Applications to Networks & Graphs	183
A.1.9 Advanced Topics	184
A.1.10 Notation & Conventions	185
A.2 Graph theory	186
A.2.1 Graphs as Models for Complex Systems	186
A.2.2 Fundamental Definitions and Notation	187
A.2.3 Substructures in Graphs	188
A.2.4 Connectivity and Distance Measures	189
A.2.5 Types of Graphs	190
A.2.6 Graph Metrics	191
A.2.7 Structural Equivalence and Matching	194
A.2.8 Trees, Traversability, and Optimization	195
A.3 Mathematical Foundation of MASFENON	197
A.3.1 Graph Model and Notation	197
A.3.2 Perturbation Propagation Model	200
A.3.3 Dissipation and Conservation Dynamics	202
A.3.4 Inter-agent Communication Mechanism	204
A.3.5 Correction Strategies: Normalization, Dissipation, Con- servation	205
A.3.6 Iterative Propagation Algorithm	208
A.3.7 Complexity Analysis	209
A.3.8 Temporal Modeling and Inter/Intra Decomposition . . .	212
A.3.9 Inter-Agent Communication Modeling	215
A.3.10 Robustness Mechanisms and Correction Models	216
A.3.11 Convergence Analysis and Stability Conditions	217
A.3.12 Numerical Stability of the Pseudoinverse	220
List of terms	221

Introduction

Understanding complex dynamic systems, particularly interconnected networks, is an increasingly critical challenge in modern scientific research. Networks, ranging from biological systems to social interactions and technological infrastructures, are frequently characterized by their intricate interactions, evolving dynamics, hierarchical structure, and nonlinear behaviors. The study of these systems often involves complex network theory and chaos theory, where subtle changes in initial conditions can lead to vastly different outcomes, making predictions extremely challenging. Traditional methodologies, such as classical static models and compartmental approaches, typically fall short in accurately capturing the rich granularity of individual interactions and temporal variability inherent to real-world networks. Consequently, sophisticated simulation techniques capable of precisely modeling these systems' dynamic evolution have become critically necessary.

In biological research, accurately modeling cell-to-cell communication networks holds profound implications for medical science, biotechnology, and fundamental biology. Cells communicate dynamically through complex signaling networks, metabolic interactions, and direct physical contacts, resulting in temporally evolving cellular behaviours critical to developmental processes, immune responses, and pathological conditions such as cancer. Current modeling methodologies often simplify these interactions into static or averaged representations, neglecting temporal characterization, spatial complexity, and interaction granularity. Therefore, computational frameworks must go beyond traditional models by capturing detailed structural interactions and integrating rich, temporally resolved data that reflect the dynamic evolution of these systems.

To address these complex challenges, I developed the Multi-Agent Adaptive Simulation Framework for Evolution in Networks of Networks (MASFENON). This innovative simulation framework explicitly aims to model and analyze dynamic processes within interconnected systems through a temporal, multi-layered approach. MASFENON uniquely combines principles from multi-agent systems, complex network theory, and chaos theory, enabling it to represent detailed interactions both within and between network layers dynamically and accurately. Its capacity to manage dynamic intra- and inter-layer communications renders it particularly effective for

capturing complexities inherent to real-world systems, including emergent phenomena, non-linear interactions, and adaptive behaviors that traditional methods struggle to represent accurately.

The necessity for substantial computational resources is a significant hurdle in accurately simulating complex dynamic systems. The computational complexity involved in simulating interactions within extensive interconnected networks demands high-performance computing (HPC) solutions. To overcome this, MASFENON leverages advanced distributed and parallel computing methodologies, integrating Message Passing Interface (MPI) for efficient inter-agent communication, and optimized linear algebra routines provided by Armadillo and OpenMP for rapid, scalable intra-layer computations. These robust computational techniques enable MASFENON to conduct large-scale simulations effectively, facilitating precise modeling across various scales, from small biological networks to vast epidemiological simulations involving thousands of interacting agents.

The adaptability and scalability of MASFENON have been validated across multiple domains. In biological research, the framework has demonstrated unprecedented resolution in modeling cellular communication through integration with single-cell RNA sequencing (scRNA-seq) data. Similarly, it has proven highly effective in simulating epidemiological phenomena, accurately replicating well-established compartmental models such as SIR, SIS, SEIR, and SEIRS while offering greater temporal resolution and granularity. This capability significantly impacts public health strategies, resource allocation, and policy development by providing more accurate predictions and enabling detailed scenario testing.

In addition to biological and epidemiological applications, MASFENON's temporal and hierarchical modeling capabilities apply to a diverse array of fields, including social dynamics, economic interactions, traffic flow management, and organizational behavior analysis. To further enhance usability and accessibility, ongoing developments in the MASFENON framework include sophisticated visualization tools and intuitive web interfaces, enabling broader user interaction and easier interpretation of complex simulation results. Furthermore, efforts toward streamlined deployment on HPC systems, cloud platforms, and distributed infrastructures are currently in progress, ensuring efficient resource utilization and accessibility.

Research Question: This Ph.D. research was initiated with the ambitious objective of creating a digital twin of cell-to-cell communication, leveraging modern sequencing technologies such as single-cell RNA sequencing (scRNA-seq), alongside advanced network modeling and algorithmic strategies. The core aim was to enable detailed, accurate simulations of cellular behaviors, facilitating predictive modeling that closely mirrors real-world biological processes. Specifically, the research sought to:

- Develop a computational framework capable of accurately modeling dynamic cellular interactions.
- Integrate high-resolution, temporally-resolved sequencing data.
- Implement advanced network algorithms and modeling techniques to simulate realistic biological processes.
- Create a flexible and adaptable simulation environment suitable for diverse biological contexts.
- Establish good practices for managing and conducting extensive computational projects, enhancing reproducibility, modularity, and maintainability.

Research Scope: Positioned at the intersection of bioinformatics, systems biology, and distributed computing, this research addresses significant current limitations in the modeling and simulation of complex biological systems. By developing a versatile computational framework, this project aims to provide a powerful tool capable of capturing the detailed dynamics of biological networks, which are crucial for accurate and predictive biological research. Moreover, the research contributes substantially to distributed computing by addressing the computational challenges associated with large-scale biological simulations, improving resource utilization, parallel computing efficiency, and computational scalability.

Research is particularly relevant within the current scientific panorama, as accurate modeling of cell-to-cell interactions and network dynamics is vital for advancements in personalized medicine, therapeutic interventions, and systems biology. Thus, this research directly addresses current challenges and offers robust methodologies that enhance the capabilities and effectiveness of bioinformatics and computational biology. In addition, the project emphasizes methodological rigor, structured research design, comprehensive documentation, and reproducible computational workflows as essential good practices for doctoral research in computational science.

Research Methodology: The project was conducted by initially focusing explicitly on the biological context of cell-to-cell communication, gradually evolving into the development of an independent, versatile simulation framework: MASFENON. This evolution reflects a methodological shift towards creating a generalized computational solution capable of simulating various complex interconnected systems beyond the initial biological focus.

The methodology adopted emphasizes the architectural design and computational implementation of the framework, stressing its high flexibility and adaptability. The development involved extensive use of distributed and parallel programming paradigms, specifically targeting high-performance computing (HPC) environments. Key aspects of the methodology include:

- Architectural design that prioritizes modularity and adaptability.

- Distributed and parallel implementation using MPI, Armadillo, and OpenMP.
- Rigorous validation through extensive simulations and benchmarking.
- Development of sophisticated visualization and web-based interfaces for enhanced user interaction and interpretation of results.
- Emphasis on good development practices, including version control, rigorous testing, code review, comprehensive documentation, and reproducibility.

These methodological choices ensure that the developed framework is robust, scalable, and applicable across various scientific domains requiring precise and dynamic simulations of complex systems.

In this thesis, we present a comprehensive account of the theoretical underpinnings, methodological advancements, rigorous validation, and broad-ranging applications of the MASFENON framework. Through detailed performance benchmarks and diverse case studies, we demonstrate how MASFENON significantly advances our ability to understand, predict, and manage complex dynamic systems. Ultimately, this work establishes a solid foundation for future research and practical applications, highlighting the critical importance of precise, scalable, and temporally resolved simulations in complex network science.

This thesis is organized into six core chapters followed by a comprehensive mathematical appendix and technical documentation. In Chapter 1, we provide the necessary background and related work, beginning with an overview of sequencing techniques and cell-to-cell communication, and proceeding to foundational concepts in network theory and multi-agent systems. Chapter 2 offers a critical evaluation of current simulation models, positioning MASFENON within the broader landscape of frameworks and identifying its key innovations. In Chapter 3, we present the MASFENON framework in depth, describing its design, implementation, and computational considerations. This chapter includes both conceptual modeling and practical engineering decisions for deployment in distributed HPC environments.

Chapter 4 focuses on experimental results, including performance benchmarking, epidemiological simulations and biological simulations, with detailed case studies of cell communication. Chapter 5 outlines potential extensions and future work, especially in the context of exascale computing and integration with biological data pipelines. Chapter 6 showcases the versatility of MASFENON by presenting use cases beyond biology, social systems, and economic modeling. We end the main body of the thesis with a conclusion, outlining the final results of the project and having some words on what was learned and how this project will impact future studies.

To support the theoretical and computational foundations, Appendix A includes formal mathematical background, covering linear algebra, graph theory, and the full derivation of the mathematical engine behind MAS-

FENON.

This structure reflects the progression of the Ph.D. work: from identifying research gaps, designing a novel approach, implementing it in an HPC context, to validating it across multiple domains of application.

Another resource to take into account is the repository at <https://github.com/josura/MASFENON>, the issues in the repository, and the correlated documentation at <https://josura.github.io/MASFENON/>, containing practical documentation on the framework's configuration, deployment, software, and usage for future developers, contributors, and researchers.

Chapter 1

Background and related work

The development of the MASFENON framework was not conceived in isolation; rather, it emerged from a convergence of multiple research directions, conceptual inspirations, and computational needs that spanned both biological modeling and high-performance computing (HPC). This chapter presents the foundational background that contextualized and shaped the design and implementation of MASFENON. It serves not only to review the current state of the art, but also to illuminate the motivations, limitations, and domain-specific challenges that originally inspired the project.

Understanding why MASFENON was necessary requires a multifaceted exploration across sequencing technologies, biological systems modeling (like cell-to-cell communication and how to model the interaction between different cells with the data that is provided), network, graph theory and computational network science, and finally multi-agent simulation. Each of these domains contributed core conceptual or methodological elements to the framework. The study of dynamic interactions in biological systems, particularly cell-to-cell communication and signaling cascades, revealed a crucial need for temporally aware, structurally flexible simulation platforms. At the same time, limitations in existing simulation tools, especially when deployed on large-scale or HPC infrastructures, highlighted the absence of frameworks that could scale with biological realism and computational demand.

This chapter is thus structured to progressively build the logic and thoughts behind MASFENON. We begin with an overview of sequencing techniques, which directly influence the types of data that biological simulations must handle, and how to model this data, since that's one of the core points of every simulation framework. We then explore how cell-to-cell communication has been computationally modeled, and why its dynamic, network-based nature requires more expressive frameworks. From there, we introduce the mathematical and algorithmic principles of complex networks, emphasizing the structural and temporal features that MASFENON was designed to support. We still don't go too much in graph theory since that will

be seen specifically in the appendix A.2, and we focus specifically on real network application and usages, in the context of why talking about networks is really important for MASFENON origin. Finally, we delve into the theory and implementation of multi-agent systems (MAS), a paradigm that inspired MASFENON’s agent-centered and communication-focused architecture, while also understanding why HPC and MAS go hand in hand. Multi-Agent Systems (MAS) consist of autonomous entities, known as agents, which interact within a shared environment to achieve collective or individual goals. In the realm of high-performance computing (HPC), MAS offer an essential approach for simulating large-scale, complex interactions efficiently. HPC environments leverage parallel and distributed computing techniques to handle computationally intensive MAS simulations, effectively addressing issues related to scalability, computational overhead, and large dataset management.

Throughout this background, we also examine the shortcomings of widely used tools, such as NetLogo[92] and NDLib[79], as well as traditional epidemiological models, particularly with respect to scalability, flexibility, and temporal granularity. NetLogo is widely used towards educational and research-oriented modeling, but struggles significantly with scalability, limiting its applicability to large-scale HPC simulations, and NDLib, a Python-based library designed for network diffusion processes, offers robust modeling capabilities but encounters limitations when scaling to very large networks or integrating fine-grained temporal dynamics. The critique that we will be bringing is not simply comparative; it highlights the practical and conceptual gaps that motivated the creation of MASFENON. We will see in depth and focus on the comparative side and how MASFENON places itself among these tools in chapter 2.

In short, this chapter does more than survey the landscape; it reconstructs the foundational decisions that guided the research, implementation, and evolution of the MASFENON project. It demonstrates how a synthesis of biological insight, computational limitations, and formal modeling created the foundations of a framework that is both general-purpose and domain-aware, scalable and adaptable.

1.1 Overview of sequencing techniques

The advent of high-throughput sequencing technologies has fundamentally transformed the fields of genomics, transcriptomics, and systems biology. Originally developed to decode entire genomes, sequencing methodologies have since evolved into a diverse ecosystem of specialized techniques, each tailored to specific biological questions, sample types, and resolution needs. Among these, bulk RNA sequencing (bulk RNA-seq) has long served as the standard for transcriptomic profiling at the tissue level, providing a macro-

scopic view of gene expression averaged across populations of cells. However, the growing awareness of cellular heterogeneity within complex tissues and the emergence of precision medicine have catalyzed a transition toward high-resolution methods such as single-cell RNA sequencing (scRNA-seq), which offer unprecedented granularity into cell-specific behaviors.

This section provides an overview of the major sequencing strategies relevant to the modeling of cellular behavior, with emphasis on both their historical progression and technological milestones. Particular attention is given to the methodologies, challenges, and interpretive frameworks associated with bulk and single-cell RNA sequencing. These technologies not only underpin many current systems biology approaches but have also had a direct impact on the conceptual foundations of the MASFENON framework.

Indeed, the early stages of MASFENON were deeply influenced by prior research conducted during my Master’s thesis, which focused on computational methods for single-cell transcriptomics and their applications in precision medicine. That work, particularly the development of the C2C-SEPIA prototype, explored the feasibility of modeling cell-to-cell communication by integrating pathway knowledge and scRNA-seq-derived expression dynamics. Many of the core design principles that now define MASFENON, such as time-resolved network simulation, agent-based perturbation propagation, and multi-layered graph modeling, can be traced back to those foundational investigations. As such, sequencing technologies were not merely sources of input data, but were instrumental in shaping the logic, architecture, and simulation paradigms upon which MASFENON was built.

By establishing this technical and intellectual lineage, this section bridges the gap between biological data acquisition and computational simulation, framing sequencing not just as a topic to consider during the initial steps of every pipeline of work with this kind of data, but as a conceptual driver in the design of dynamic modeling systems.

1.1.1 Bulk sequencing

Bulk RNA sequencing (bulk RNA-seq) has been a cornerstone of transcriptomic research since the early 2000s, following the achievements of the Human Genome Project. It enables comprehensive quantification and characterization of RNA populations within complex biological samples, providing valuable insights into gene expression, alternative splicing, and transcriptomic dynamics. Applications span diverse fields, including cancer biology[37, 95, 38], inflammation research[17, 98], and autoimmune disease studies[59, 66].

Experimental Workflow Overview A typical bulk RNA-seq experiment consists of several key experimental and computational steps:

1. **Sample Collection and RNA Isolation:** Biological material (e.g., tissue biopsies) is collected, and total RNA is extracted. Maintaining RNA integrity and minimizing contamination are critical at this stage. Commercial kits (e.g., Qiagen, ThermoFisher) are often employed[83].
2. **Library Preparation:** The extracted RNA undergoes several sub-steps:
 - **RNA selection or depletion:** Poly(A) selection to enrich mRNA, or rRNA depletion for broader transcript detection[73, 47].
 - **Reverse transcription and cDNA synthesis:** Conversion of RNA into cDNA for sequencing.
 - **Fragmentation and Size Selection:** Optimization of fragment sizes, typically between 200-500 bp.
 - **Adapter Ligation:** Addition of sequencing adapters containing barcodes and UMIs.
 - **Amplification:** PCR or linear amplification to enrich libraries, with UMIs minimizing amplification bias[44, 40].
3. **Sequencing:** Libraries are sequenced using platforms like Illumina NovaSeq or HiSeq. Sequencing can be performed as single-end or paired-end, depending on the experimental goals[39].

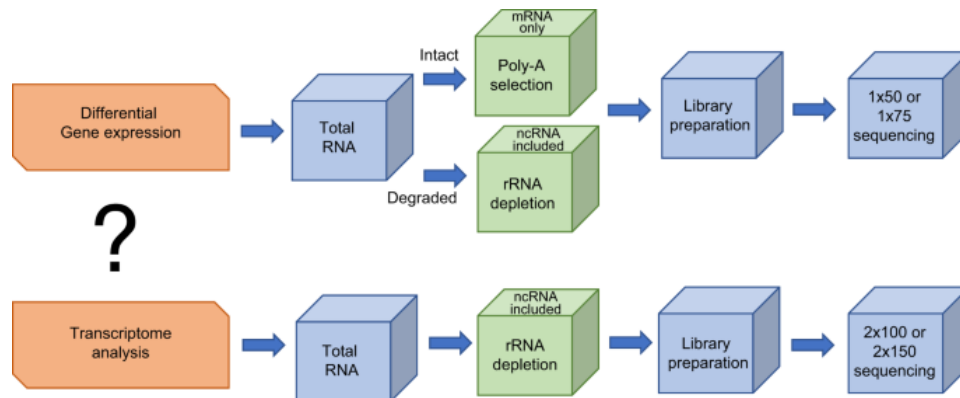


Figure 1.1: Two types of bulk RNA-seq library preparation workflows: poly(A) selection and rRNA depletion.

Computational Analysis Workflow After sequencing, bulk RNA-seq data analysis involves several algorithmic and statistical steps:

1. **Read Preprocessing and Quality Control:** Raw reads (FASTQ files) are assessed using tools like FastQC. Adapter trimming and quality filtering may be applied.
2. **Alignment or Pseudo-alignment:**
 - **Alignment:** Reads are mapped to a reference genome or transcriptome using aligners like STAR[26].
 - **Pseudo-alignment:** Tools like Salmon[70] estimate abundances

without full alignment using lightweight mapping algorithms.

3. Quantification:

- **Gene or Transcript Counting:** Using featureCounts[48] or HTSeq[3].
- **Normalization:** Application of RPKM, FPKM, or TPM methods to control for sequencing depth and transcript length[106].

4. Differential Expression Analysis:

- DEGs are identified using statistical models (e.g., DESeq2[53], edgeR[77]).
- Multiple testing corrections (e.g., Benjamini-Hochberg FDR) are applied.

5. Downstream Analyses:

- Functional enrichment using GO, KEGG.
- Visualization: PCA, heatmaps, volcano plots.

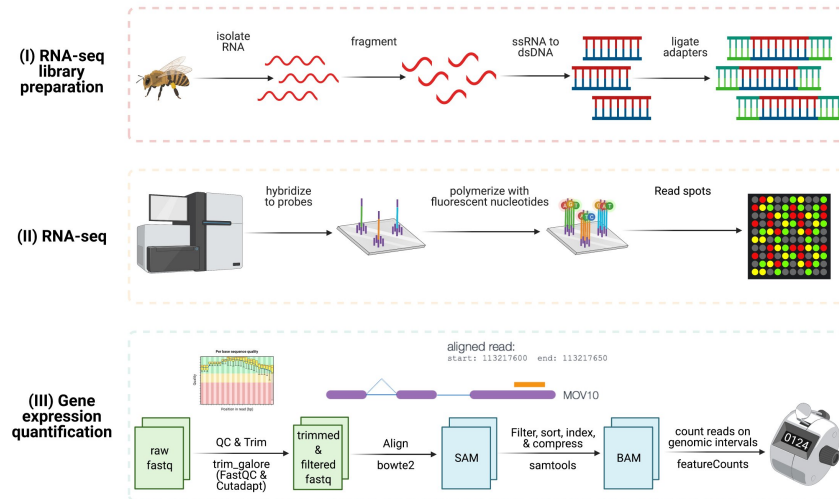


Figure 1.2: General workflow of bulk RNA-seq experiments, from RNA isolation to differential expression analysis.

Strengths and Limitations Bulk RNA-seq offers several advantages:

- **Reproducibility:** Well-established and standardized protocols.
- **Cost-effectiveness:** Suitable for large cohort studies.
- **Depth of coverage:** Enabling robust detection of low-abundance transcripts.

However, it also presents notable limitations:

- **Cellular Averaging:** Masks cellular heterogeneity within complex tissues.
- **Loss of Resolution:** Cannot distinguish between cell-type-specific transcriptional programs.

These limitations have driven the development of single-cell RNA sequencing (scRNA-seq) technologies, which offer higher-resolution insights into the dynamic behavior of individual cells[68, 85].

1.1.2 Single cell sequencing

Single-cell RNA sequencing (scRNA-seq) has revolutionized transcriptomic analysis by enabling the profiling of gene expression at the resolution of individual cells. Unlike bulk RNA-seq, which produces an averaged transcriptional profile, scRNA-seq captures the complexity of cellular heterogeneity, uncovering rare populations, dynamic processes, and novel cellular states. This technology plays a crucial role in developmental biology, cancer research, immunology, and systems biology.

Recent years have witnessed an explosion of new protocols, computational methods, and integrated multi-omics approaches. Single-cell transcriptomics now intersects with fields such as spatial biology, epigenomics, proteomics, and dynamic modeling of cellular fate decisions. In this chapter, we present the principles, methodologies, computational pipelines, and challenges associated with scRNA-seq, with an emphasis on both classical methods and the latest technological advances.

Objectives and Challenges The primary objectives of scRNA-seq are:

- Identification of distinct cell types and subpopulations.
- Characterization of transcriptional heterogeneity across tissues and conditions.
- Reconstruction of developmental and differentiation trajectories.
- Profiling of cellular responses to genetic or environmental perturbations.

Despite its transformative potential, scRNA-seq faces several challenges:

- **Technical Noise:** Dropouts and low capture efficiency complicate downstream analyses.
- **High Dimensionality:** Thousands of genes per cell require dimensionality reduction and noise handling.
- **Batch Effects:** Variations due to experimental conditions must be corrected.
- **Cost and Scalability:** Full-length sequencing remains costly for large-scale experiments.

Experimental Workflow Overview The general workflow for scRNA-seq includes:

1. **Sample Preparation:** Tissue dissociation or nuclei isolation.
2. **Single-Cell Capture:** FACS, micromanipulation, nanowells, or droplet microfluidics.

3. **Library Preparation:** Poly(A) capture, reverse transcription, amplification, and barcoding (UMIs and cell barcodes).
 4. **Sequencing:** Illumina short-read platforms are most commonly used.
 5. **Data Processing:** Demultiplexing, alignment or pseudoalignment, quantification, normalization, clustering, and downstream analysis.
- Figure 1.3 illustrates the general workflow.

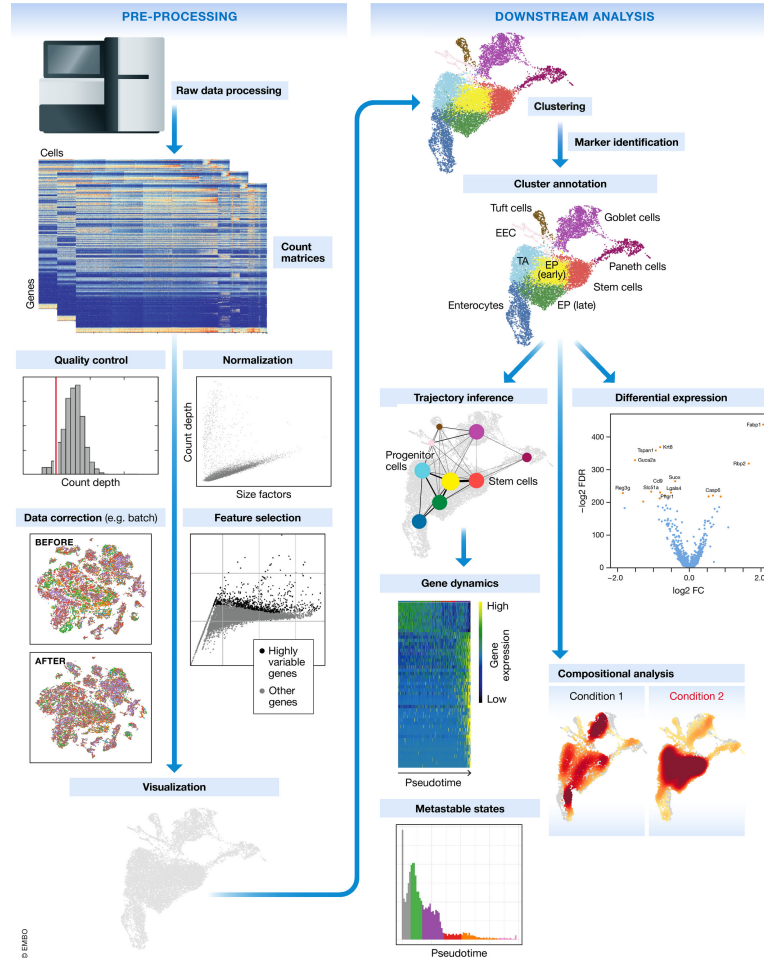


Figure 1.3: General workflow of scRNA-seq.

Technological Advances Recent developments have expanded the capabilities of scRNA-seq:

- **Multi-Omics Integration:** Protocols such as CITE-seq [86] and SHARE-seq allow the simultaneous measurement of RNA expression with surface proteins or chromatin accessibility[56]. Computational tools like Seurat v5 and MOFA+ facilitate data integration across omics layers.

- **RNA Velocity and Dynamic Modeling:** Methods like scVelo[8] and DeepVelo infer future states of cells by modeling spliced and unspliced RNA kinetics, expanding the ability to reconstruct developmental trajectories. This type of capability is also what MASFENON is built as well and on of the main focus of the framework (the dynamic modeling and the evolution of a system of cells).
- **Spatial Transcriptomics:** Technologies like LCM-seq[63], tomo-seq[46], and Geo-seq[18] preserve the spatial context of gene expression, bridging single-cell analysis with tissue architecture. This is also one of the future objectives for MASFENON, as we will see in chapters 5 and 6
- **Deep Learning and Large Language Models:** Deep generative models (e.g., scVI[52]) and graph neural networks (e.g., GraphSCI[76]) improve denoising, clustering, and imputation of single-cell data. Recent work also applies large language models (LLMs) to automate cell type annotation based on known markers, streamlining analysis workflows. Other deep learning model can also infer new modalities from scRNA-seq data (e.g. scFEA [2])

Comparison of Protocols Protocols can be broadly categorized into:

- **Full-Length Profiling** (e.g., Smart-seq2[72]): High sensitivity, suitable for isoform detection, but lower throughput.
- **3'/5' Tag-Based Profiling** (e.g., 10x Genomics Chromium[33]): High throughput, cost-effective, UMI-based counting.
- **Split-Pool Barcoding** (e.g., SPLiT-seq[78]): Extremely scalable, no need for physical isolation.

Trade-offs exist between throughput, sensitivity, and cost, and should be considered based on the experimental goal.

Computational Analysis Pipeline The analysis of single-cell transcriptomic data typically follows a well-defined computational pipeline, designed to transform raw sequencing reads into biologically interpretable insights. Each step addresses specific challenges such as noise, technical variability, and high dimensionality, ultimately enabling the identification of meaningful cellular states and dynamics. A representative pipeline can be summarized as follows:

1. **Preprocessing:** Initial quality control ensures that only reliable cells and features are retained for downstream analysis. This includes filtering out low-quality reads, doublets, and dead or stressed cells, as well as removing genes expressed in very few cells.
2. **Normalization:** Expression levels must be corrected for sequencing depth and other technical effects. Standard approaches include log-normalization or more advanced variance-stabilizing methods such as SCTransform[81], which improve comparability across cells and condi-

tions.

3. **Dimensionality Reduction:** To address the inherent high dimensionality of scRNA-seq data, methods like principal component analysis (PCA) are applied to capture major sources of variation, followed by nonlinear techniques such as UMAP[61] or t-SNE for visualization in low-dimensional space.
4. **Clustering:** Cells are grouped into putative cell types or states using graph-based clustering methods such as Louvain or Leiden. These algorithms operate on nearest-neighbor graphs constructed from reduced-dimensional representations, allowing detection of both large and fine-grained subpopulations.
5. **Differential Expression:** Marker genes distinguishing clusters are identified through statistical testing. This step enables biological interpretation by linking computationally identified clusters with known cell types or functional states.
6. **Trajectory Inference:** For dynamic processes such as differentiation or response to stimuli, trajectory inference methods (e.g., Monocle3[45]) order cells along pseudotime, reconstructing likely developmental or signaling pathways.
7. **Imputation and Denoising (optional):** To mitigate dropout events and technical noise, imputation methods such as MAGIC[24], DeepImpute, or scImpute can be applied. These tools estimate missing expression values, improving downstream analysis in cases where sparsity is a limiting factor.

This pipeline provides the foundation for more advanced integrative analyses, such as cell-cell communication modeling or multimodal data integration, which build directly on the outputs of the steps above.

Applications scRNA-seq is applied in:

- Uncovering new cell types.
- Studying development and differentiation.
- Investigating tumor heterogeneity and immune landscapes.
- Exploring spatial tissue organization.
- Precision medicine and drug response prediction.

Additional Methods :

Smart-seq3[34] extends Smart-seq2 by incorporating UMIs into a full-length transcript sequencing protocol, addressing amplification biases while enabling allele-resolved expression and isoform discovery. Although the protocol is complex, Smart-seq3 is a major milestone toward high-sensitivity, full-length transcriptomics at single-cell resolution.

Advanced Computational Steps. Beyond the standard preprocessing and normalization procedures, single-cell analysis relies on a set of advanced computational steps that enable the identification of cellular subpopulations, the exploration of dynamic processes, and the generation of biologically interpretable visualizations.

Unsupervised clustering, such as hierarchical clustering, k-means, and graph-based methods, is a central component of scRNA-seq analysis. Among these, graph-based approaches (e.g., Seurat[82], which employs Louvain or Leiden community detection) have become the gold standard due to their ability to capture complex relationships in high-dimensional single-cell data and to detect both broad cell classes and subtle subpopulations.

Following clustering, cell groups are typically **annotated** through differential expression analysis to identify marker genes. Manual annotation leverages domain expertise, while automated methods such as SingleR[84] accelerate the process by comparing cluster-specific profiles to curated reference datasets, thereby reducing subjectivity and increasing reproducibility.

Dimensionality reduction methods (PCA, UMAP, t-SNE) are employed both to mitigate noise and to enable interpretation of high-dimensional data. PCA provides linear projections that capture variance but often fails to represent nonlinear biological structures. By contrast, manifold learning approaches such as UMAP[61] or t-SNE better preserve local neighborhoods and global topology, making them particularly well-suited for scRNA-seq visualization and interpretation of fine-grained cellular relationships.

Visualization plays a crucial role in the interpretability of single-cell analysis. Common practice involves projecting cells into 2D or 3D space using UMAP or t-SNE, where clusters and trajectories can be intuitively inspected. However, visualization extends beyond simple embedding: latent space representations derived from autoencoders or variational models can provide richer, denoised embeddings that capture higher-order structures in the data. Furthermore, visualizations can be augmented with overlays of metadata (e.g., timepoint, cell type, experimental condition) to highlight biological trends and batch effects.

Trajectory analysis is a powerful step for reconstructing dynamic biological processes such as cell differentiation, activation, or disease progression. Algorithms like Monocle3[45], Slingshot[89], or PAGA[99] order cells along a pseudotime axis, effectively arranging them according to their progression through a biological continuum. This enables the identification of branch points corresponding to lineage bifurcations, as well as temporal gene expression patterns driving cellular transitions. Integration with RNA velocity methods can further enhance trajectory inference by incorporating directionality into the pseudotime ordering, providing insights into not only the order of states but also the dynamics of their evolution.

Taken together, these advanced computational steps transform high-dimensional and noisy scRNA-seq data into biologically interpretable rep-

representations, enabling the discovery of novel cell types, the reconstruction of dynamic processes, and the generation of testable biological hypotheses.

Applications and Impact :

Single-cell sequencing technologies have transformed fields such as cancer research, immunology, and regenerative medicine. Precision medicine increasingly relies on scRNA-seq to resolve tumor heterogeneity, predict drug responses, and guide therapy selection. The ability to dissect the "cellulome" of tissues enables the discovery of novel therapeutic targets and biomarkers at unprecedented resolution.

Challenges and Considerations Despite remarkable progress, scRNA-seq faces several challenges:

- **Sparsity and Dropouts:** Low capture efficiency requires careful normalization and imputation.
- **Batch Effects:** Technical variability must be corrected with proper experimental design and computational methods.
- **Doublets and Multiplets:** Especially in droplet-based methods, where two cells may share a barcode.
- **Loss of Spatial Context:** Although spatial methods are emerging, most scRNA-seq protocols lose positional information.
- **Complex Data Integration:** Single-cell multi-omics and large-scale datasets necessitate robust integration frameworks.

Figure 1.4 illustrates the importance of avoiding confounding experimental designs.

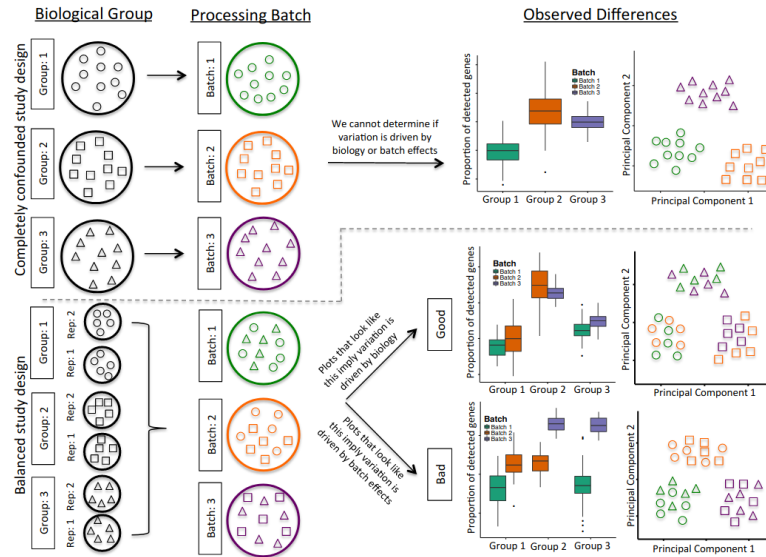


Figure 1.4: Illustration of a confounded (top) versus balanced (bottom) design. Source[36].

1.1.3 Comparison between Bulk and Single-Cell RNA Sequencing

While single-cell RNA sequencing (scRNA-seq) has revolutionized transcriptomic analysis by offering insights into the heterogeneity of complex biological systems, it is crucial to contextualize its advancements relative to traditional bulk RNA sequencing (bulk RNA-seq) technologies. Understanding the differences, strengths, and limitations of both approaches allows for a more nuanced appreciation of their complementary roles in biological research.

Bulk RNA-seq, widely adopted after the development of next-generation sequencing (NGS) technologies, measures the average gene expression across an entire population of cells. It has been successfully applied to a wide range of studies, from identifying differentially expressed genes between healthy and diseased tissues to characterizing evolutionary transcriptomic changes across species. Bulk RNA-seq's strengths lie in its maturity, cost-effectiveness, and the depth of information it provides about whole tissues or homogeneous samples. However, because it measures averaged expression across millions of cells, it inherently obscures cellular heterogeneity, masking the presence of rare cell types, intermediate states, or dynamic transitions between phenotypes.

In contrast, scRNA-seq isolates and profiles the transcriptomes of individual cells, offering an unparalleled resolution into cellular diversity. First introduced in 2009 [91], scRNA-seq became broadly accessible around 2014 with the development of improved protocols and the reduction of sequenc-

ing costs. Unlike bulk methods, scRNA-seq reveals distributions of gene expression at the single-cell level, enabling the discovery of novel cell types, rare populations, and fine-grained characterization of cellular differentiation processes. It is now fundamental for constructing high-resolution cell atlases and investigating biological phenomena that require the understanding of heterogeneity, such as immune responses, developmental pathways, and tumor evolution.

A detailed comparison of the two approaches, including their objectives, workflows, normalization strategies, and analytical pipelines, is summarized in Table 1.1.

	Goal	Protocol	Quality Control	Normalization	Analyses
Bulk RNA-seq	• Measure average gene expression across a population of cells • Identify differences between sample conditions	• RNA extracted from mixed cell populations • Reverse transcription and sequencing library construction • Amplification and sequencing	• GC content biases, adaptor contamination • Mapping rates and duplication levels • Reproducibility across replicates	• Between-sample normalization: quantile normalization, spike-ins • Within-sample normalization: RPKM, FPKM, TPM	• Differential expression analysis • Alternative splicing and isoform quantification • Transcriptome assembly
scRNA-seq	• Measure gene expression at single-cell resolution • Identify novel cell types, subpopulations, and states	• RNA captured from individual cells • Labeling with unique molecular identifiers (UMIs) • Reverse transcription, amplification, and library preparation	• Cell and gene filtering based on mitochondrial content and library size • Read quality, UMI collapsing	• Specialized methods for dropout correction and capture bias • Normalization of UMI counts or read counts (e.g., CPM, scran)	• Dimensionality reduction and clustering • Cell-type annotation and biomarker discovery • Trajectory and pseudotime analysis

Table 1.1: Comparison between bulk RNA sequencing and single-cell RNA sequencing workflows and analyses

Although scRNA-seq unlocks the ability to resolve cellular heterogeneity, it introduces its own technical challenges, including dropout events, amplification biases, and data sparsity, requiring sophisticated computa-

tional methodologies for normalization, imputation, and interpretation, as discussed earlier in this chapter. Conversely, bulk RNA-seq remains the method of choice for homogeneous tissues and studies requiring high sequencing depth, reproducibility, and cost-efficiency.

Each methodology has its ideal application context: bulk RNA-seq is most suitable for profiling relatively homogeneous populations or obtaining an averaged signal across a tissue, whereas scRNA-seq is indispensable when cellular heterogeneity, rare populations, or dynamic cell-state transitions are the key focus.

Importantly, emerging strategies now seek to **combine bulk and single-cell RNA-seq data** to capitalize on the strengths of both approaches. Single-cell data can inform the deconvolution of bulk expression profiles, while bulk data can improve the robustness of gene expression estimation and imputation in scRNA-seq, as detailed in recent studies [58, 42, 71].

The synergy of these approaches defines the future of transcriptomic profiling, enabling a more comprehensive and accurate understanding of biological systems. For a more comprehensive review of sequencing technologies and pipelines, and a connection to this Ph.D. thesis, see my Master’s thesis at <http://dx.doi.org/10.13140/RG.2.2.35352.06408>

1.2 Overview of cell-to-cell communication

Cell-cell interaction (CCI) and cell-cell communication (CCC) are fundamental processes governing the organization, function, and evolution of multicellular systems. Before meaningful molecular communication can occur between cells, an initial physical or chemical **interaction** must take place, establishing the foundation for signaling events. Recent advances in bioinformatics, particularly with single-cell omics technologies, have provided unprecedented means to computationally infer both CCI and CCC at high resolution, revealing the complex networks that underlie tissue development, immune responses, and disease progression.

In the context of networks, biological network modeling aims to represent and analyze complex interactions within and between biological entities, crucial for understanding phenomena like signaling pathways, metabolic regulation, and cellular differentiation. Cell-to-cell communication is particularly essential for deciphering multicellular dynamics, developmental biology, immune responses, and disease progression. Recent advances in single-cell RNA sequencing (scRNA-seq) and related techniques provide temporally resolved data, highlighting the need for frameworks like MASFENON that can effectively utilize these data types to simulate dynamic interactions accurately. Traditional static or compartmental models frequently lack the temporal granularity necessary to fully capture biological processes, reinforcing the necessity of temporally resolved, network-based simulations as provided by

MASFENON.

1.2.1 Cell-Cell Interaction (CCI)

Cell-Cell Interaction (CCI) refers to the establishment of physical or proximity-based relationships between two or more cells, which create the structural and spatial preconditions for communication. These interactions determine whether and how molecular signals can be transmitted between cells.

Cells can interact through several modes:

- **Contact-dependent interactions:** Involving direct physical contact via adhesion molecules (e.g., cadherins, integrins), tight junctions, desmosomes, or gap junctions. These interactions facilitate not only mechanical stability but also direct transfer of ions or small molecules.
- **Extracellular Matrix (ECM) mediation:** Cells interact indirectly through a shared ECM substrate, which provides both mechanical anchoring and biochemical signaling cues (mechanotransduction, integrin signaling).
- **Nanotube-mediated contact:** Thin membranous tubes connecting distant cells, allowing exchange of ions, vesicles, or even organelles.
- **Spatial proximity:** Cells located near each other may interact through diffusion-based signaling (paracrine) or localized field effects, even in the absence of direct contact.
- **Mechanical coupling:** Cells transmit mechanical stress or deformation through the ECM or direct adhesion points, influencing neighboring cell morphology and gene expression.

Understanding CCI is crucial, as the feasibility and specificity of downstream **cell-cell communication (CCC)** depend on whether cells are positioned and connected appropriately. Not all theoretically possible ligand-receptor interactions are biologically realized; spatial organization and adhesion often gate communication potential.

With the rise of spatial transcriptomics and single-cell imaging, computational frameworks such as **Giotto** [29] and **SpaOTsc** [15] integrate spatial and transcriptomic data to infer potential CCIs within tissues.

1.2.2 Cell-Cell Communication (CCC)

Once structural interactions or proximity are established, cells engage in active communication, the transfer of molecular or physical signals that modulate behavior, fate, and function. CCC thus represents the *functional layer* built upon the interaction topology defined by CCI.

Cell-cell communication governs essential biological processes, including development, immune response, tissue homeostasis, and disease progression. Traditionally studied through microscopy or biochemical assays, CCC can now be computationally inferred from high-throughput single-cell and spa-

tial omics data, particularly by analyzing **ligand-receptor pairs** between interacting cell populations.

1.2.3 Mechanisms of Cell-Cell Communication

Communication between cells can occur through several distinct mechanisms, which can be broadly categorized as follows:

- **Juxtacrine signaling:** Contact-dependent communication via membrane-bound ligands and receptors (e.g., Notch-Delta pathway).
- **Gap junctional exchange:** Direct cytoplasmic continuity between neighboring cells via connexins, allowing passage of ions, metabolites, and small molecules.
- **Paracrine signaling:** Local communication where secreted ligands act on nearby cells.
- **Autocrine signaling:** Cells secrete ligands that act back on their own receptors.
- **Endocrine signaling:** Long-distance signaling through hormones or other circulating molecules that travel via the bloodstream.
- **Synaptic signaling:** Fast, spatially precise signaling between neurons through neurotransmitter release across synaptic clefts.
- **Mechanical signaling:** Transmission of mechanical forces or ECM tension that modulate signaling cascades (e.g., via mechanosensitive channels).
- **Electrochemical coupling:** In excitable tissues (e.g., cardiac muscle), coordinated electrical activity spreads through gap junctions, synchronizing function.

In computational biology, inference of CCC primarily focuses on **paracrine**, **autocrine**, and **juxtacrine** signaling, detectable through transcriptomic patterns of ligands and receptors. Advanced frameworks integrate spatial constraints from CCI to increase the biological plausibility of predicted signaling events. Different type of intercellular interactions can be seen in Figure 1.5

1.2.4 Computational Inference of Cell-Cell Communication

The computational analysis of CCC or CCI involves several key steps:

1. **Identification of cell types or states:** Clustering scRNA-seq data to define distinct cellular populations.
2. **Detection of expressed ligands and receptors:** Assessing the expression levels of known ligand and receptor genes within each cell type.
3. **Inference of interactions:** Predicting potential communication events based on co-expression of ligand-receptor pairs between sender and receiver cell types.

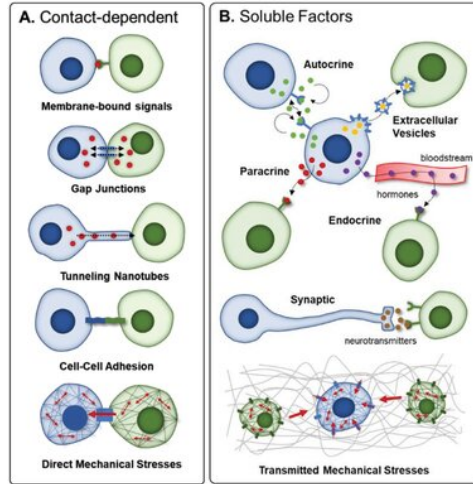


Figure 1.5: Different types of intercellular interactions. Source [102]

4. **Inferring spatial proximity or adhesion likelihood** (for CCI).
5. **Quantification and visualization:** Scoring the strength of interactions and representing the communication network through various visualization techniques.

Several bioinformatics tools have been developed to facilitate these analyses:

- **Giotto:** Integrates spatial data with single-cell expression to infer CCIs and downstream CCC. Nowadays, the inference of CCC and CCI are integrated in **Giotto Suite**, which is an open-source software package designed for spatial data analysis in the R environment. It offers a technology-agnostic data framework for representing and integrating diverse spatial datasets, along with scalable and extensible solutions for data analysis, integration, and visualization. Giotto Suite enables the integration of molecular, morphology, spatial, and annotated feature information to create a responsive and flexible workflow for multi-scale, multi-omic data analyses. An image representing the capabilities of Giotto suite can be seen in Figure 1.6
- **CellPhoneDB:** Utilizes a curated database of ligand-receptor interactions to predict CCC networks from scRNA-seq data, accounting for subunit architecture and complex formation [93]. CellphoneDB is an available repository of manually curated and reviewed molecular interactions with evidenced roles in cellular communication. It can be used to interrogate single-cell transcriptomics data and identify potential cell-cell interactions. The database includes only interactions involving HUMAN receptors, ligands, and their subunit architectures, as well as biosynthetic pathways and non-protein molecules acting as ligands. For an image of the pipeline, see figure 1.7.

- **CellChat**: Implements a probabilistic model to infer and analyze CCC networks, identifying major signaling inputs and outputs for each cell group [43]. CellChat is an open-source R package that infers, visualizes, and analyzes intercellular communications from single-cell RNA-seq data. It models the probability of cell-cell communication by integrating gene expression with prior knowledge of the interactions between signaling ligands, receptors, and their cofactors. CellChat can operate in label-based and label-free modes, and provides several visualization outputs and quantitative analyses to facilitate intuitive user-guided data interpretation. With the current iteration of CellChat v2, built upon the foundations of CellChat v1, offers expanded capabilities, including the ability to analyze multiple spatially resolved transcriptomics datasets and infer spatially proximal cell-cell communication. CellChat v2 also includes an updated ligand-receptor database, new functionalities for interfacing with other computational tools, and an interactive web browser for exploring CellChat outputs. Cellchat also differentiates from other tools since the inference of the Ligand-receptor interactions is not a 1-to-1 relation, but a **many-to-1** relation with more than 1 ligand and one receptor. To see an overview of Cellchat, see Figure 1.8. We will use **CellChat** to infer the Ligand-Receptor interaction in section 4.3.
- **NicheNet**: NicheNet is a computational method for inferring ligand-target regulatory potential in the context of intercellular communication, with a particular focus on how signaling molecules affect gene expression in target (receiver) cells. It stands apart from many other tools in the cell-cell communication (CCC) field by not limiting itself to identifying active ligand-receptor pairs. Instead, it integrates a comprehensive prior model of intracellular signaling and gene regulatory networks to predict the downstream transcriptional effects of ligands on target genes. At its core, NicheNet combines three main data layers: a ligand-receptor interaction network, a signaling and regulatory pathway network connecting receptors to target genes, and observed gene expression data from sender and receiver cells (e.g., scRNA-seq clusters). The framework uses these components to compute ligand-target regulatory potential scores, allowing researchers to rank ligands not just by their presence or expression, but by their likely influence on the transcriptional program of receiving cells. The output includes ranked ligand candidates, predicted target genes per ligand, and prioritized ligand-receptor pairs. This ability to couple ligand activity with gene expression changes offers mechanistic insights into how specific cell types influence others through signaling, particularly useful in developmental biology, tumor microenvironment studies, and immune modulation scenarios. For example, in the original publication [14], NicheNet was successfully applied to prioritize ligands involved

in monocyte activation and fibroblast-T cell crosstalk, highlighting its biological relevance in complex multicellular systems. To see the general workflow of NicheNet, see Figure 1.9 and look at the repository at <https://github.com/saeyslab/nichenetr>

- **ICELLNET**: Focuses on quantifying CCC by considering the expression of multi-subunit ligand-receptor complexes, offering a versatile framework for various biological contexts [60]. ICELLNET infers CCC from human transcriptomic data. It is an R package that uses a manually curated database of ligand-receptor interactions to compute communication scores between cells. To access and use ICELLNET, there is a repository available in GitHub at <https://github.com/soumelis-lab/ICELLNET>. Compared to other tools, ICELLNET’s strength lies in its comprehensive expert curated database of experimentally demonstrated human interactions and its ability to consider the multiple subunits of ligand-receptor complexes. To see a pipeline of how the tool works, see Figure 1.10
- **scConnect**: Provides an exploratory analysis of CCC by constructing interaction networks based on ligand-receptor expression and assessing their significance through permutation testing [41]. ScConnect is both a method and a software tool for analyzing single-cell RNA sequencing data to detect interactions between cell types based on their gene expression profiles. It infers ligand-receptor interactions and constructs a multi-directional graph, enabling contextual exploratory analysis. ScConnect is open source and available as a Python package, and it can be used with Scanpy pipelines. It is different from other tools as it specifically focuses on inferring molecular ligand-receptor interactions and constructing a multi-directional graph for exploratory analysis. For more detailed information on how to use ScConnect, refer to the documentation and tutorial available on its GitHub repository at <https://github.com/JonETJakobsson/scConnect>. To see a pipeline of how the tool works

While most CCC inference methods primarily focus on ligand-receptor co-expression and their downstream signaling effects, newer approaches extend the analysis to **metabolic interactions**. One notable example is **scFEA** (single-cell Flux Estimation Analysis)[2], which integrates single-cell transcriptomic data with a genome-scale metabolic network to infer cell-wise metabolic flux distributions. By coupling gene expression with stoichiometric constraints, scFEA estimates the metabolic states of individual cells and identifies how intercellular signaling influences metabolic activity. This integration allows researchers to investigate not only which cells communicate, but also how such communication alters metabolic pathways and cross-cell metabolic dependencies.

Incorporating tools such as scFEA into CCC analyses provides an additional layer of biological insight, enabling the reconstruction of *signaling-*

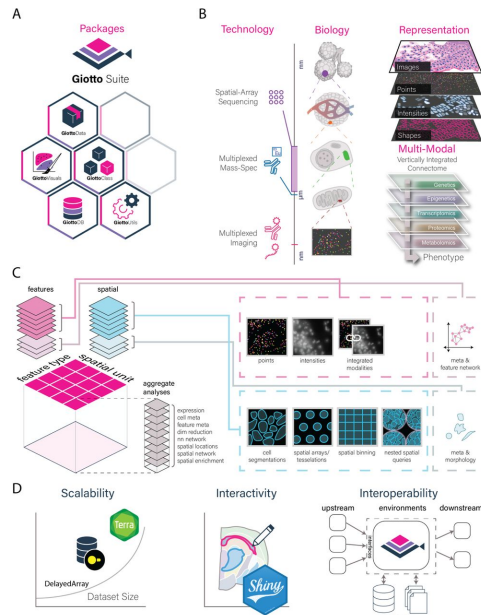


Figure 1.6: Giotto Suite ecosystem and core data framework. For more information about the suite, visit the documentation <https://giottosuite.readthedocs.io/en/latest> and the article [16].

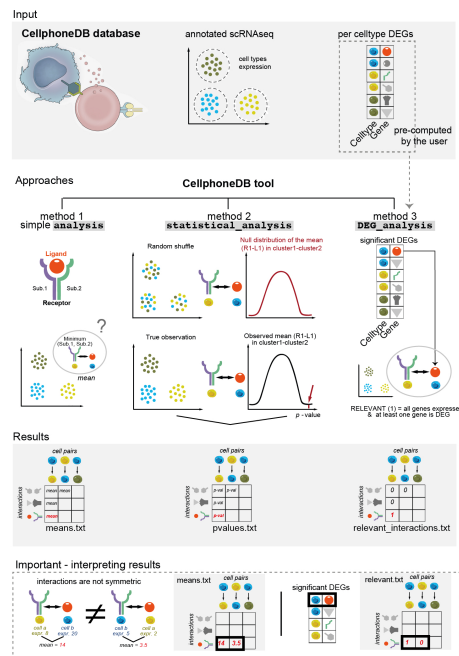


Figure 1.7: CellPhoneDB pipeline, for more information visit the documentation at <https://cellphonedb.readthedocs.io/en/stable/RESULTS-DOCUMENTATION.html>

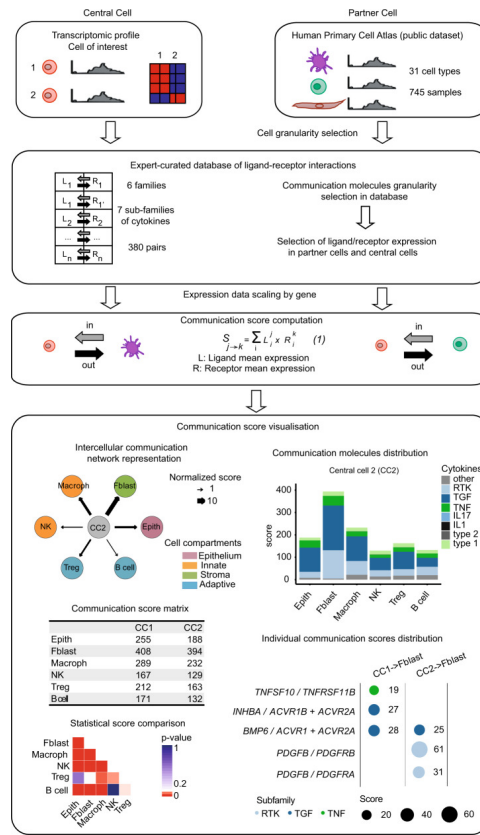


Figure 1.10: ICellNet Pipeline. For more information, see source [64].

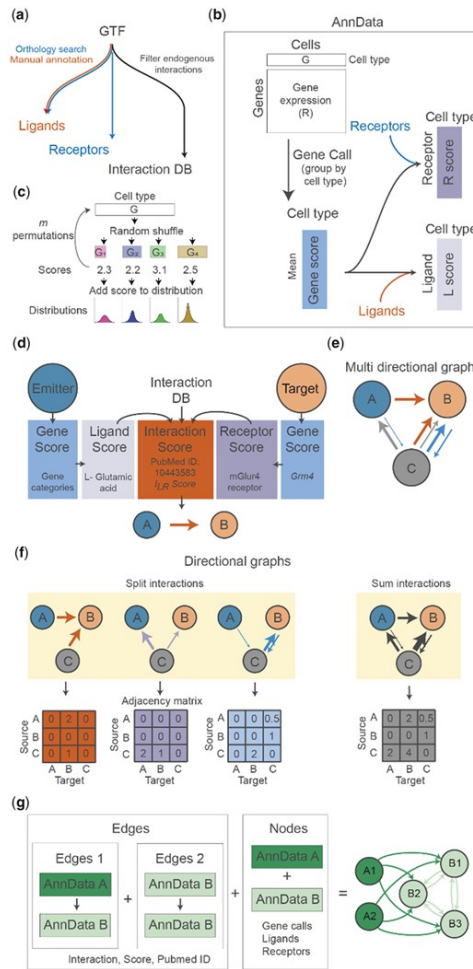


Figure 1.11: ScConnect Overview. For more information, see source [41]

metabolism crosstalk. This is particularly relevant in contexts such as tumor microenvironments or immune responses, where cell-cell communication is tightly coupled to metabolic reprogramming. By combining ligand-receptor-based frameworks (e.g., CellChat, NicheNet) with metabolic modeling approaches like scFEA, it becomes possible to capture a more holistic view of cellular interactions, spanning from surface-level communication to intracellular metabolic outcomes.

1.2.5 Applications in Health and Disease

Computational CCC analysis has been instrumental in various biological and medical research areas:

- **Cancer research:** Uncovering interactions between tumor cells and the microenvironment, aiding in the identification of therapeutic targets.
- **Immunology:** Understanding immune cell communication in response to infections or vaccinations.
- **Developmental biology:** Mapping signaling pathways during embryogenesis and tissue differentiation.
- **Neurology:** Investigating neuron-glia interactions and their roles in neurodegenerative diseases.
- **Regenerative medicine:** Informing strategies for tissue engineering by elucidating communication networks that promote tissue repair.

1.2.6 Challenges and Future Directions

Despite significant advancements, several challenges persist in CCC analysis:

- **Incomplete interaction databases:** Current ligand-receptor databases may lack comprehensive coverage, leading to potential false negatives.
- **Spatial context:** Traditional scRNA-seq lacks spatial information, which is crucial for accurate CCC inference. Integration with spatial transcriptomics is an emerging solution.
- **Dynamic interactions:** Capturing temporal changes in CCC during processes like development or disease progression remains complex.
- **Validation:** Computational predictions require experimental validation to confirm biological relevance.

Future developments aim to address these challenges by enhancing databases, integrating multi-omics data, incorporating spatial and temporal dimensions, and developing standardized benchmarking frameworks [4].

An important step toward overcoming these challenges has been the development of **Liana** and its extension **Liana+** [25]. Liana provides a unified framework that integrates and benchmarks multiple state-of-the-art CCC inference methods and experimental databases (e.g., CellPhoneDB, CellChat, Connectome), allowing users to compare outputs in a standardized manner.

and mitigate method-specific biases. Its strength lies in its ability to harmonize diverse algorithms under a common interface, making large-scale and reproducible CCC analysis more accessible.

Building on this, **Liana+** extends the framework to incorporate novel technologies such as spatial transcriptomics and multimodal datasets and inferencing (MetalinksDB[30], that we will also use in section 4.3), offering enhanced scalability and improved handling of context-specific interactions. Together, these tools exemplify the trend toward standardized, modular, and integrative platforms that will play a central role in the next generation of CCC analyses, bridging methodological advances with real-world biological complexity.

1.2.7 Conclusion

Cell-cell interaction is the essential prerequisite for cell-cell communication, setting the stage for coordinated tissue behavior. Advances in bioinformatics and single-cell technologies have enabled unprecedented insights into these intercellular relationships, with profound implications for developmental biology, immunology, cancer research, and regenerative medicine. As datasets and methods mature, computational models of CCI and CCC will continue to deepen our understanding of complex biological systems.

Integrating bioinformatics approaches with high-throughput sequencing technologies has revolutionized the study of cell-cell communication. Computational tools enable the systematic inference of intercellular signaling networks, providing insights into the complex interplay of cells within tissues. Continued advancements in this field hold promise for deepening our understanding of biological systems and informing therapeutic interventions.

1.3 Overview of networks

1.3.1 Introduction to Networks Across Domains

The concept of a network, a collection of discrete entities connected by relationships, serves as a unifying abstraction across a vast range of scientific domains. In its most basic form, a network is composed of nodes (also referred to as vertices) representing entities, and edges (or links) representing interactions or relationships between them. This deceptively simple structure has proven remarkably powerful in modeling the organizational principles of natural, engineered, and social systems.

From the neural circuits of *C. elegans* to global transportation infrastructures, networks allow researchers to capture the complex interplay between local interactions and global phenomena. In computer science, networks form the foundation of the internet, peer-to-peer systems, and routing protocols. In sociology, they enable the study of interpersonal relationships, information

diffusion, and community structure. In physics and engineering, networks are used to model power grids, supply chains, and failure propagation. In epidemiology, contact networks guide the modeling of disease spread, while in economics and organizational science, networks of firms, consumers, or institutions reflect the interdependence of economic actors.

Notably, biology has emerged as one of the most fertile domains for network-based thinking. Networks offer a compelling way to model molecular interactions, such as gene regulation, protein-protein interactions, and metabolic pathways, as well as higher-level cellular processes, including cell signaling and communication. At each of these levels, the network paradigm allows researchers to abstract the system’s structure and behavior in terms of connectivity, flow, modularity, and dynamics.

Despite their diversity, these systems often exhibit recurring structural patterns and universal properties that transcend their specific domains. Features such as small-world topology, heavy-tailed degree distributions, and modular community structure are frequently observed across domains, suggesting the presence of general principles that govern the organization of complex systems [6, 62].

As networks become increasingly central to modern data-driven science, the need to simulate and analyze them in dynamic, heterogeneous, and large-scale settings grows accordingly. A major objective of this thesis is to explore these complexities across domains while focusing in particular on biological systems, where the interplay of structure and dynamics is both subtle and consequential. However, to accommodate the wide range of potential applications, the simulation framework presented here **MASFENON** is designed not only with biological networks in mind but also with the flexibility to operate across disciplinary boundaries.

1.3.2 Fundamental Network Properties

Networks across domains can be systematically analyzed through key topological measures that originate from graph theory (for more information about the formal definition of networks and other metrics, see Appendix A.2), we will only see an abstraction of these metrics since they are already described in detail in the appendix. These metrics quantify local and global structural properties and are central to understanding the behavior of dynamical processes on networks.

Degree and Degree Distribution. The degree of a node is defined as the number of edges connected to it. In directed networks, this is separated into in-degree and out-degree. The distribution of node degrees across the network, known as the degree distribution, is a key descriptor of network topology. Many real-world networks exhibit heavy-tailed degree distributions, including power-law distributions in scale-free networks, where a small

number of nodes (hubs) have a disproportionately large number of connections [6]. This property often reflects underlying growth mechanisms like preferential attachment.

The degree distribution $P(k)$ is the fraction of nodes with degree k . Many real-world networks follow a power-law:

$$P(k) \sim k^{-\gamma}$$

with $2 < \gamma < 3$ [6].

Path Length and Diameter. The path length between two nodes is the number of edges in the shortest path connecting them. The average path length across all node pairs gives insight into how quickly information or influence can propagate through the network. The diameter of a network is the longest of all shortest paths.

The shortest path $d(u, v)$ is the minimum number of edges needed to travel from u to v . The average shortest path length is:

$$L = \frac{1}{|V|(|V| - 1)} \sum_{\substack{u, v \in V \\ u \neq v}} d(u, v)$$

The graph diameter is the longest shortest path:

$$D = \max_{u, v \in V} d(u, v)$$

Many natural and artificial networks are small-world, meaning they exhibit short average path lengths even as they grow in size. Small-world networks exhibit low L and high clustering [96].

Clustering Coefficient. The clustering coefficient quantifies the tendency of a node's neighbors to be connected to each other. It reflects the presence of tightly-knit groups or motifs in the network and is often higher in real-world networks than in random graphs. Local clustering coefficients can be aggregated into a global measure to compare overall network cohesiveness. You can see the formal definition of local and global clustering coefficients in the appendix A.2

The global clustering coefficient is defined as:

$$C = \frac{\text{number of closed triplets}}{\text{number of all triplets}}$$

Community Structure and Modularity. Networks often exhibit modular organization, in which nodes are densely connected within groups (communities or modules) but sparsely connected between them. Community

detection algorithms, such as Louvain, aim to identify these modules and are particularly useful in domains like biology, where communities might correspond to functional gene groups, signaling pathways, or interacting cell types.

Modularity Q evaluates the strength of division of a network into communities:

$$Q = \frac{1}{2m} \sum_{u,v \in V} \left[A_{uv} - \frac{\deg(u) \deg(v)}{2m} \right] \delta(c_u, c_v)$$

where A_{uv} is the adjacency matrix, c_u is the community of u , and $m = |E|$ [62].

Centrality Measures. Centrality measures identify influential or structurally important nodes:

- Degree centrality: nodes with many connections.
- Betweenness centrality: nodes that frequently occur on shortest paths between other nodes.
- Closeness centrality: nodes that can reach others with few hops.
- Eigenvector centrality: nodes connected to other highly connected nodes.

These metrics are used in various contexts, such as identifying regulatory genes, super-spreaders in epidemics, or key routers in internet traffic. To see these metrics in details, refer to appendix A.2

Motifs and Subgraph Frequencies. Motifs are small subgraphs $H \subset G$ found at significantly higher frequency than expected by chance. The significance of motif H is assessed via a Z-score:

$$Z_H = \frac{N_H(G) - \langle N_H(G_{\text{rand}}) \rangle}{\sigma(N_H(G_{\text{rand}}))}$$

where G_{rand} is a random network preserving the degree distribution.

In biological networks, motifs like feedforward loops or feedback inhibition patterns often correspond to regulatory circuits or robust control mechanisms. The detection of motifs provides insight into the basic building blocks of complex networks.

Assortativity. Assortativity measures the tendency of nodes to connect to others that are similar in some property, such as degree. Biological networks tend to be disassortative, with hubs often connecting to low-degree nodes. Some networks also display hierarchical structure, where small, densely connected modules nest within larger ones, a trait especially relevant to systems like metabolic or organizational networks.

The assortativity coefficient r measures the correlation of degrees between connected nodes:

$$r = \frac{\sum_{uv}(k_u - \bar{k})(k_v - \bar{k})}{\sum_{uv}(k_u - \bar{k})^2}$$

Positive r indicates assortative mixing; negative r indicates disassortativity, common in biological networks.

These foundational properties not only allow networks to be quantitatively characterized and compared but also serve as constraints or targets in the modeling and simulation of dynamic processes and are crucial for understanding network function, robustness, and emergent behavior. More formal definitions and proofs are presented in AppendixA.2, including further discussion of walks, cycles, Laplacians, and spectral properties. In the context of MASFENON, many of these properties can be tuned, preserved, or evolved depending on the simulation goals, particularly for the analysis of multi-scale biological systems or the reproduction of empirically observed topologies and behaviour, since, depending on these metrics, the simulation will produce similar behaviour.

1.3.3 Types and Representations of Networks

Understanding the structural diversity of networks is essential before applying modeling or simulation approaches. Networks encountered in real-world systems, ranging from biological to technological domains, can exhibit a wide range of topologies. These typologies often emerge from different generative principles and significantly affect dynamics on networks.

1. Random Graphs (Erdős-Rényi Model). The Erdős-Rényi (ER) model $G(n, p)$ is one of the simplest network models. Given n nodes, each of the $\binom{n}{2}$ possible edges is independently included in the graph with probability p . The degree distribution follows a binomial law:

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}$$

and converges to a Poisson distribution for large n and small p . Although analytically tractable, ER graphs lack high clustering and heavy-tailed degree distributions observed in many real systems.

2. Scale-Free Networks (Barabási-Albert Model). Many real-world networks (especially biological ones) exhibit power-law degree distributions:

$$P(k) \sim k^{-\gamma}, \quad \text{with } 2 < \gamma < 3$$

The Barabási-Albert (BA) model generates such networks through preferential attachment. Starting with a small seed graph, new nodes are added

iteratively and attach to existing nodes with probability proportional to their degree:

$$\Pi(v_i) = \frac{\deg(v_i)}{\sum_j \deg(v_j)}$$

This "rich-get-richer" mechanism leads to hubs, which have a disproportionate influence on network dynamics, resilience, and communication.

3. Small-World Networks (Watts-Strogatz Model). The Watts-Strogatz (WS) model interpolates between regular lattices and random graphs to capture small-world properties: high clustering and short average path length. Starting from a ring lattice with k -nearest neighbors, edges are rewired with probability β . As $\beta \rightarrow 0$, the network is regular; as $\beta \rightarrow 1$, it becomes random.

These networks support efficient local communication and are commonly observed in neural and social systems.

4. Modular and Hierarchical Networks. Modular networks are organized into communities, subsets of nodes with dense internal connections and sparser interconnections. Hierarchical networks exhibit nested communities, with structure across multiple scales.

Such modularity is fundamental in systems biology (e.g., functional gene modules) and organizational networks. Formally, modularity is captured by the measure Q introduced earlier, and hierarchical clustering can be performed recursively.

5. Bipartite and Multipartite Graphs. A bipartite graph $G = (U, V, E)$ consists of two disjoint sets U and V , with edges only between nodes from different sets. These are common in biological networks: for example, gene-disease or ligand-receptor interactions.

Extensions include multipartite graphs and hypergraphs, where edges may connect more than two nodes. Bipartite projections often result in loss of structural fidelity, making direct modeling preferable.

6. Temporal and Dynamic Networks. Unlike static graphs, temporal networks encode time-varying interactions. These can be represented as a sequence of snapshots $\{G_t\}$, or more compactly as time-stamped edges (u, v, t) .

Such representations are critical for simulating processes like signal propagation, epidemic spreading, or dynamic cell-to-cell communication. Formal modeling of these systems is discussed in detail in the MASFENON methodology section and the temporal network appendix.

7. Multiplex and Multi-Layer Networks. In multi-layer networks, the same set of nodes can interact across different layers representing different interaction types. Formally, a multiplex network is defined as:

$$\mathcal{G} = \{G^{[1]}, G^{[2]}, \dots, G^{[L]}\}$$

where $G^{[l]} = (V, E^{[l]})$ is the graph at layer l .

These are particularly relevant for systems where different kinds of interactions coexist, such as genetic, metabolic, and signaling pathways in biological cells. Inter-layer couplings are often weighted, and interdependencies can significantly affect global dynamics.

8. Weighted and Signed Networks. In many real-world systems, interactions are not binary. Weighted graphs assign a value $w(u, v) \in \mathbb{R}$ to each edge, representing strength, frequency, or capacity.

Signed networks allow $w(u, v) \in \{-1, 1\}$ or general real values to represent activating or inhibiting interactions (e.g., in gene regulation or trust/distrust in social networks). Signed Laplacians and spectral methods are used to analyze such systems (see AppendixA.2).

9. Spatial and Geometric Networks. In spatial networks, nodes occupy positions in Euclidean or metric space, and edge formation depends on geometric constraints. Examples include road networks, protein-protein interactions constrained by folding geometry, or neuronal connectomes.

Graphs may be embedded using spatial coordinates $x_i \in \mathbb{R}^d$, and distances affect edge probability via kernel functions:

$$P((i, j) \in E) \propto f(\|x_i - x_j\|)$$

Summary. These network typologies provide the foundational vocabulary for modeling complex systems, including biological, social, and technological domains. These network types also dictate the behaviour of the whole network in the context of perturbation and evolution dynamics of the network, as we will also see is the case for MASFENON as well in section 4.2 in the context of epidemics, when the type of synthetic network used also influences directly the behaviour of the system and its components. While many networks can be characterized using a combination of the above types, each introduces specific constraints and properties that must be accounted for in modeling and simulation. A formal treatment of their mathematical properties and generative models can be found in AppendixA.2.

1.3.4 Modeling Real-World Networks

Real-world networks, including biological, social, economic, and technological systems, are typically derived from empirical observations. These observations often come in the form of complex, noisy, and incomplete datasets,

requiring careful methods for network modeling. The process of transforming empirical data into a structured network model includes multiple steps: data collection, preprocessing, node and edge identification, and model validation.

Data Collection and Preprocessing. Networks constructed from empirical data usually originate from experimental observations, database mining, surveys, sensor recordings, or high-throughput technologies. For instance, in biological systems, gene expression data from bulk RNA-seq or scRNA-seq can reveal co-expression or regulatory relationships; similarly, epidemiological models often start from contact tracing and patient interaction data. Initial preprocessing typically includes data cleaning, normalization, and handling missing values or outliers.

Node and Edge Identification. A critical stage in network construction involves defining nodes and edges, dependent on the system under study. In biological networks, nodes might represent genes, proteins, cells, or organisms, while edges could represent interactions such as binding events, signaling, or gene regulatory influences.

Edges can be identified through various methodologies, for example:

- **Correlation-Based Methods:** A common practice in gene co-expression networks is to define edges using correlation coefficients (e.g., Pearson, Spearman) between expression vectors. Formally, an edge between nodes i and j can be defined by a correlation threshold:

$$E(i, j) = \begin{cases} 1 & \text{if } \rho(i, j) \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where $\rho(i, j)$ is the correlation between nodes and θ is a user-defined threshold.

- **Regression and Machine Learning-Based Methods:** Methods like regression modeling or mutual information can capture directed regulatory relationships. For instance, in gene regulatory networks, a directed edge $X_i \rightarrow X_j$ indicates regulation from gene X_i to gene X_j . Regression models typically use:

$$X_j = \sum_{i \neq j} \beta_{ij} X_i + \epsilon_j$$

where significant coefficients β_{ij} define edges.

- **Physical or Experimental Validation:** In protein-protein interaction (PPI) networks, edges may derive directly from experimental validations (e.g., yeast two-hybrid assays, mass spectrometry, chromatin immunoprecipitation sequencing (ChIP-seq)).

Network Representation: Adjacency and Incidence Matrices. After identifying nodes and edges, networks are represented mathematically through structured matrices.

The simplest representation is an adjacency matrix A , which for a network $G = (V, E)$ with n nodes, is defined as:

$$A_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

For weighted networks, each edge carries a weight w_{ij} , and the adjacency matrix is given by:

$$A_{ij} = \begin{cases} w(i, j) & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

The incidence matrix B , particularly useful in flow-based models, is defined for a directed network with nodes n and edges m as:

$$B_{ie} = \begin{cases} -1 & \text{if edge } e \text{ leaves node } i \\ +1 & \text{if edge } e \text{ enters node } i \\ 0 & \text{otherwise} \end{cases}$$

Such representations simplify analytical manipulation and computational modeling.

Fitting Generative Network Models. Real-world network data are often compared with generative network models (e.g., Erdős-Rényi, Barabási-Albert, Watts-Strogatz) to understand underlying formation processes or validate model accuracy. This process typically involves parameter estimation to maximize model-data agreement. For example, fitting a Barabási-Albert model involves determining the attachment parameter m from observed degree distributions.

Network Validation and Robustness. Constructed networks must be validated to ensure biological or empirical accuracy. Common validation strategies include:

- **Cross-validation:** Networks derived from subsamples of data should yield consistent topological and functional properties.
- **Functional enrichment:** For biological networks, network modules or clusters should align with known functional annotations or pathways.
- **Sensitivity and robustness analysis:** Small perturbations in data or parameters should not drastically alter the network structure or key network properties.

Limitations and Challenges in Real-World Network Modeling. Several inherent limitations and challenges complicate the modeling of empirical networks, including:

- **Data Quality:** Real datasets are noisy, incomplete, and often contain systematic biases.
- **Scale and Complexity:** High-dimensional data (such as omics data) pose computational and algorithmic challenges.
- **Temporal and Dynamic Changes:** Many real-world systems are not static, necessitating temporal models or dynamic networks.

Despite these challenges, careful modeling provides critical insights into the structural and functional principles underlying complex systems.

Connecting to MASFENON. The methodological approaches outlined here serve as a necessary foundation for the development of the MASFENON framework. Accurate modeling of empirical networks ensures fidelity and applicability of network-based simulations to biological and other interconnected systems. Details on the specific methodological implementations in MASFENON, including how network topology affects propagation dynamics and multi-agent interactions, will be extensively explored in subsequent chapters on the framework methodology, especially in Chapter 3.

For an in-depth mathematical background, refer to Appendix A.2.

1.3.5 Networks Dynamics and Flow Processes

Understanding the dynamics of processes evolving over networks is fundamental to modeling complex systems, including biological, epidemiological, social, and technological domains. In network science, *network dynamics* refers to the mechanisms by which information, signals, matter, or agents propagate or diffuse across the underlying network structure.

Types of Dynamic Processes in Networks. Network dynamics broadly include several categories of processes, each exhibiting distinct behavior:

- **Diffusion Processes:** Represented mathematically by diffusion or random walks, where entities such as molecules or information move randomly from node to node. Diffusion dynamics are commonly used to model signaling pathways in cells or the spread of information in social networks.
- **Epidemiological Processes:** Classic compartmental models like SIR (Susceptible-Infectious-Recovered) or SIS (Susceptible-Infectious-Susceptible) describe the spread of pathogens. These models incorporate node states and transition probabilities between compartments to represent disease transmission.
- **Synchronization and Oscillatory Dynamics:** Nodes in certain networks exhibit synchronized or oscillatory behavior, such as neurons

firing collectively or circadian gene expression cycles. Kuramoto-type models are commonly employed to model synchronization.

- **Flow and Transport Processes:** Networks can model the movement of resources, such as blood flow in vascular networks, energy distribution in power grids, or traffic flow in transportation networks. These processes are typically described using flow conservation equations.

Mathematical Formulation of Network Dynamics. The dynamics on networks are typically formulated through systems of ordinary differential equations (ODEs), partial differential equations (PDEs), difference equations, or stochastic models, depending on the level of detail and nature of the process.

A general continuous-time network dynamical system on a network $G = (V, E)$ can be described by:

$$\frac{dx_i}{dt} = f_i(x_i, \{x_j\}_{j \in \mathcal{N}(i)}, t)$$

where:

- x_i is the state (or concentration, information level, etc.) at node i ,
- $\mathcal{N}(i)$ is the set of neighbors of node i ,
- f_i represents the local update function depending on node i and its neighbors.

Discrete-time processes can be modeled similarly using difference equations:

$$x_i(t+1) = f_i(x_i(t), \{x_j(t)\}_{j \in \mathcal{N}(i)}, t)$$

Diffusion and Random Walk Dynamics. A foundational model for network diffusion is the random walk. The evolution of the probability distribution $\mathbf{p}(t)$ over the nodes at time t is given by:

$$\mathbf{p}(t+1) = \mathbf{W}\mathbf{p}(t)$$

where $\mathbf{W}$ is the transition matrix whose elements w_{ij} describe the probability of transition from node j to node i . For simple random walks on an unweighted network, we often use:

$$w_{ij} = \frac{A_{ij}}{\sum_k A_{kj}}$$

with A_{ij} the adjacency matrix element.

Compartmental Epidemiological Models. Epidemiological processes on networks are often represented using compartmental models. For instance, the classic Susceptible-Infectious-Recovered (SIR) model on a net-

work is defined by three compartments with the following transition dynamics:

$$\frac{dS_i}{dt} = -\beta S_i \sum_{j \in \mathcal{N}(i)} A_{ij} I_j, \quad \frac{dI_i}{dt} = \beta S_i \sum_{j \in \mathcal{N}(i)} A_{ij} I_j - \gamma I_i, \quad \frac{dR_i}{dt} = \gamma I_i$$

where S_i , I_i , and R_i represent the susceptible, infected, and recovered state proportions at node i , respectively, and parameters β and γ represent transmission and recovery rates.

Synchronization Dynamics. The Kuramoto model provides a classical description of synchronization in oscillator networks, where the state of each node i is represented by an angle (phase) θ_i :

$$\frac{d\theta_i}{dt} = \omega_i + \sigma \sum_{j=1}^n A_{ij} \sin(\theta_j - \theta_i)$$

with ω_i being the intrinsic frequency of node i , and σ the coupling strength. Network synchronization is characterized by the collective alignment of phases.

Flow Processes and Conservation Laws. Flow processes in networks are governed by principles of conservation laws, such as Kirchhoff's law in electrical circuits or conservation of mass in metabolic networks. Formally, for a node i :

$$\frac{dx_i}{dt} = \sum_{j \in \mathcal{N}(i)} F_{ji} - \sum_{j \in \mathcal{N}(i)} F_{ij} + q_i$$

where F_{ij} is the flow rate from node i to node j , and q_i represents sources ($q_i > 0$) or sinks ($q_i < 0$) at node i . These equations capture the balance of inflows, outflows, and internal production or consumption at each node.

Analyzing Network Dynamics. Key analysis methods include:

- **Stability Analysis:** Determines equilibrium points and their stability using linear stability analysis or Lyapunov functions.
- **Numerical Simulations:** Computational simulations for large-scale or complex models, critical for scenarios lacking closed-form solutions.
- **Spectral Methods:** Analysis based on network eigenvalues and eigenvectors to predict dynamical behavior and synchronization.

Relevance for MASFENON. In the MASFENON framework, network dynamics and flow processes are fundamental. The intra- and inter-layer propagation dynamics described in MASFENON rely heavily on the principles outlined here. Network dynamics guide the computational implementation, influencing how information propagates across agents within and

between layers and determining time-evolution rules. The computational specifics of these dynamic processes in MASFENON will be extensively detailed in later chapters.

1.3.6 From Data to Network: Construction Challenges

Building networks from real-world data involves a series of methodological challenges that significantly influence the accuracy, robustness, and interpretability of the resulting networks. Network construction typically requires transforming raw data, such as experimental measurements, observations, or sensor readings, into meaningful connections between nodes. However, several inherent challenges emerge from data characteristics and the methods used for inference.

Inference from Raw Data Networks can be inferred from data using various methodologies, each carrying different implications:

- **Co-occurrence methods:** This simplest form of inference involves defining edges based on the frequency of simultaneous occurrence or spatial proximity of entities. While computationally straightforward, co-occurrence often fails to capture complex relationships and causal links between nodes.
- **Correlation-based methods:** Correlation coefficients (e.g., Pearson, Spearman, or Kendall correlation) measure linear or monotonic relationships between node pairs. Edges are defined when correlations exceed predetermined thresholds. Despite their widespread use, these methods are sensitive to spurious correlations and confounding effects, often misrepresenting indirect relationships as direct interactions.
- **Causal modeling:** Techniques such as Bayesian networks, Granger causality, or dynamic causal modeling attempt to infer directional relationships, capturing causation rather than mere association. Though powerful, these approaches require extensive computational resources and strong assumptions that might not hold in all contexts.

Sparsity, Noise, and Uncertainty in Edge Definition One critical issue in constructing networks from data is managing sparsity, noise, and uncertainty inherent in most real-world datasets:

- **Sparsity:** Many real-world networks are sparse, containing relatively few edges compared to the number of potential connections. Sparse datasets challenge accurate network inference, making it difficult to distinguish true relationships from incidental ones.
- **Noise:** Experimental and observational data often contain substantial noise, obscuring genuine relationships between nodes. Techniques for noise reduction or robust statistics must be employed to mitigate its impact.

- **Uncertainty:** Edge definitions frequently involve probabilistic or uncertain criteria. Methods such as probabilistic graphical models or Bayesian approaches are beneficial to explicitly incorporate uncertainty and produce robust network representations.

Time-Resolved Data and Snapshot Construction Networks often evolve dynamically, and data is frequently collected as time series or sequential snapshots. Several issues must be addressed when transforming such data into networks:

- **Temporal granularity:** Selecting appropriate time intervals for snapshot generation is critical. Too coarse granularity may obscure rapid dynamics, while excessively fine granularity can introduce unnecessary noise and redundancy.
- **Dynamic inference:** Accurately capturing temporal relationships requires methods that consider historical states of nodes and their interactions, such as temporal graphs or dynamic Bayesian networks.

Heterogeneous Sources (Multi-omics, Social Sensors, Logs) Increasingly, network construction integrates data from heterogeneous sources, multi-omics in biological research, logs from computer systems, and social sensors in sociological studies. These heterogeneous sources pose additional complexity:

- **Data integration:** Combining datasets with different resolutions, formats, and modalities demands careful normalization and integration strategies to avoid introducing systematic biases or distortions.
- **Multi-scale representation:** Different data sources often capture phenomena at varying scales (e.g., molecular, cellular, organismal in biological contexts). Network construction approaches must reconcile these scales, enabling meaningful interaction across different data types.

Understanding these challenges is critical for accurately translating data into reliable and informative networks. Addressing them effectively lays the foundation for meaningful subsequent analyses and robust interpretations in network science.

1.3.7 Biological Networks in Focus

Networks are among the most effective representations for complex systems, allowing binary relationships between entities to be clearly modeled and analyzed. In biological contexts, entities (nodes) include biomolecules such as genes, proteins, metabolites, cells, or entire organisms, while relationships (edges) represent biochemical, regulatory, or signaling interactions. These relationships can reflect direct physical interactions, functional dependencies, or temporal events.

Biological networks integrate methods from biology, physics, computer science, and mathematics, providing comprehensive insights into complex systems. Leveraging topological features of these networks helps model systemic responses, such as drug efficacy predictions or disease progression mechanisms. Understanding network structure is crucial since biological entities often function collectively rather than independently.

Three primary categories of biological networks include:

- **Biochemical Networks:** Represent molecular interactions and control mechanisms within cells. Key biochemical networks include:
 - *Metabolic Networks:* Detail chemical transformations within metabolic pathways, capturing the flow of metabolites and energy. Nodes represent metabolites or enzymes, and edges depict chemical reactions.
 - *Protein-Protein Interaction (PPI) Networks:* Highlight physical interactions between proteins, crucial for understanding complex molecular functions.
 - *Gene Regulatory Networks (GRNs):* Illustrate how transcription factors and regulatory elements govern gene expression, thus influencing cellular processes and development.
 - *Signaling Networks:* Represent signal transduction mechanisms that communicate external stimuli into cellular responses.
- **Neural Networks:** Biological neural networks comprise interconnected neurons that facilitate sensory, motor, and cognitive processes. Technological advances (e.g., multi-electrode arrays, optical imaging) have enhanced our ability to map and understand complex neural interactions, revealing properties such as small-world organization within cortical structures.
- **Ecological Networks:** Model interactions within ecosystems, including predation, mutualism, and competition among species. Ecological networks help analyze ecosystem stability and structural complexity, which are essential for understanding ecological resilience and biodiversity.

Among biological networks, pathways play a particularly critical role, capturing sequential biochemical reactions leading to cellular changes. Pathways facilitate understanding how molecular interactions culminate in functional outcomes such as metabolism regulation, gene expression modulation, or cell signaling.

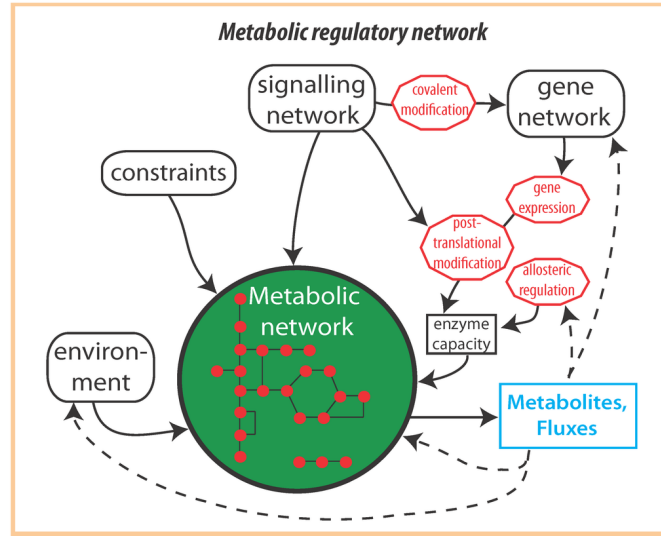


Figure 1.12: Overview of regulatory interactions within metabolic networks illustrating interactions between metabolic processes and regulatory mechanisms (source: [9]).

Metabolic networks are fundamental in personalized medicine, identifying precise metabolic disruptions in diseases and enabling targeted therapeutic strategies. Such networks help in modeling drug effects, understanding pathogen-host interactions, and revealing critical intervention points within metabolic pathways.

Gene regulatory networks (GRNs) encompass relationships among transcription factors and their target genes, representing the dynamic regulatory landscape of gene expression. Inferring GRNs from transcriptomic data (bulk and single-cell RNA-seq) supports understanding developmental processes, disease mechanisms, and responses to environmental changes. Advances in single-cell technologies have significantly enhanced the resolution and accuracy of GRN inference.

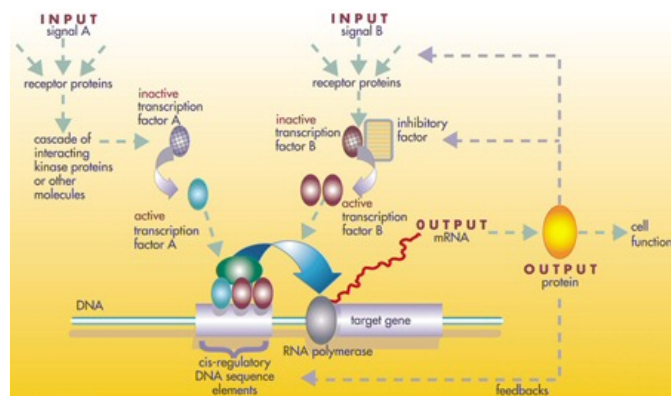


Figure 1.13: Typical structure of a gene regulatory network demonstrating regulatory interactions (source: [97]).

Signaling pathways depict mechanisms transmitting external signals into intracellular responses through cascades of biochemical reactions. Identifying active signaling pathways through differential gene expression analysis enhances understanding cellular responses and pathway-specific alterations linked to diseases.

Protein-protein interaction (PPI) networks detail the physical interactions between proteins, essential for cellular functions and signaling cascades. Experimental methods like X-ray crystallography, affinity chromatography, and yeast two-hybrid assays, along with computational inference techniques, are critical in PPI discovery.

As introduced before in section 1.1, Cell-cell interaction (CCI) networks model intercellular communication, crucial in understanding organismal development, homeostasis, and immune responses. Inferring CCIs from ligand-receptor expression profiles in transcriptomic datasets facilitates the identification of critical communication pathways influencing cellular and systemic behaviors.

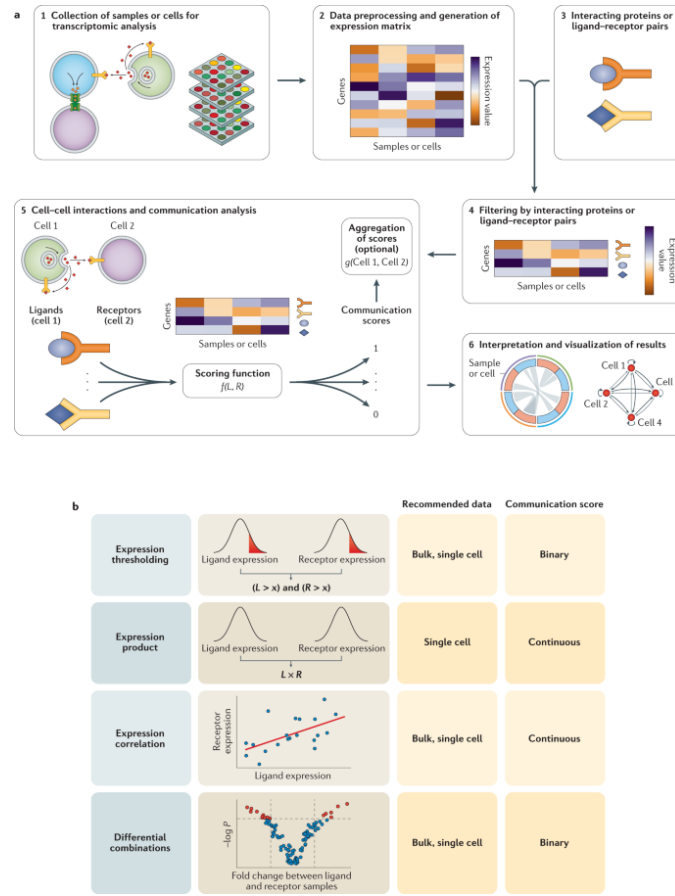


Figure 1.14: Workflow for inferring cell-cell interactions and communication from gene expression data (source: [5]).

Additionally, correlation networks derived from co-expression analysis provide insights into gene function and interactions, assisting in identifying modules of genes with common regulatory mechanisms. These networks are particularly valuable in uncovering novel biological processes and biomarkers.

Finally, biological knowledge graphs, structured using semantic triples and RDF frameworks, integrate diverse biological information. They enable cross-database queries, machine learning applications, and comprehensive multi-omic analyses, significantly enhancing data integration and biological knowledge discovery.

In summary, biological networks are essential tools across various domains of biology and medicine, providing frameworks for understanding complexity, predicting system behaviors, and designing targeted interventions.

1.3.8 Why General Network Simulation Frameworks Are Needed

Networks have emerged as a fundamental abstraction for understanding complex systems across various domains, including biology, sociology, information technology, and economics. While specialized simulation models and tools exist within these domains, the increasing complexity, heterogeneity, and interdisciplinary nature of real-world problems necessitate the development and adoption of general network simulation frameworks.

General simulation frameworks are needed for several critical reasons:

- **Interoperability and Integration:** Specialized tools are typically developed with domain-specific assumptions and constraints, often hindering their integration or interoperability with tools from other domains. General frameworks allow seamless integration of heterogeneous data sources and models, facilitating comprehensive simulations that accurately reflect complex interactions and dependencies among diverse network components.
- **Standardization and Reproducibility:** Domain-specific simulation tools often lack consistency in methodologies, data formats, and metrics, making the comparison and reproduction of results challenging. Generalized frameworks enforce standardized modeling practices, data handling, and result interpretation, greatly enhancing the reproducibility and reliability of simulations.
- **Scalability and Flexibility:** Real-world networks often involve extensive, complex interactions spanning various scales, from cellular and molecular interactions in biological systems to global-scale interactions in social and economic networks. A generalized simulation framework can dynamically scale across these diverse complexities, efficiently accommodating the demands of large-scale networks while maintaining computational feasibility.
- **Cross-Domain Insights:** Insights gained from one domain's network modeling and simulation can often be beneficially applied to another. General frameworks encourage the transfer and adaptation of methodologies and findings, enabling novel discoveries and innovative solutions across traditionally isolated fields.
- **Handling of Uncertainty and Noise:** Real-world networks are frequently subject to uncertainty, incomplete data, and noise. General simulation frameworks are specifically designed to accommodate and systematically explore these uncertainties, providing robust methods to evaluate the resilience and reliability of network behaviors under varied and uncertain conditions.
- **Support for Advanced Analytical Methods:** The versatility of general frameworks allows for the easy incorporation of advanced analytical and computational methods, such as machine learning, network embedding techniques, and dynamic modeling approaches, thus en-

hancing the depth and utility of network simulations.

- **Cost and Resource Efficiency:** Developing and maintaining domain-specific simulation tools can be resource-intensive. A general, modular framework provides a common basis for simulation, significantly reducing the overall costs associated with developing new tools for each specialized domain and allowing resources to be directed toward improving existing modules and adding novel functionalities.

In conclusion, the need for general network simulation frameworks arises from the imperative to integrate diverse information, ensure reproducibility, and foster interdisciplinary collaboration. Such frameworks not only streamline computational modeling practices but also greatly enhance our ability to understand, predict, and manage complex network-driven phenomena across multiple fields of study.

1.4 Overview of Multi-Agent Systems

Multi-agent systems (MAS) have been a central pillar of this Ph.D. project, forming the conceptual and computational foundation for simulating complex biological interactions and modeling diverse dynamical systems. Their relevance extends beyond abstraction: principles of MAS programming, together with paradigms such as reactive, distributed, and parallel programming, were essential for designing MASFENON. In particular, the need to handle high-dimensional and large-scale datasets such as scRNA-seq made MAS approaches especially suitable, providing the scalability, modularity, and adaptability required for realistic simulations.

1.4.1 Introduction to Multi-Agent Systems (MAS)

Definition of Multi-Agent Systems A Multi-Agent System (MAS) is a computational framework composed of multiple autonomous agents that interact and communicate within a shared environment to achieve individual or collective objectives [100]. Each agent is endowed with independent decision-making capabilities and can dynamically adapt its behavior in response to interactions with other agents and environmental feedback. These systems are inherently decentralized and frequently exhibit complex, emergent behaviors that cannot be directly inferred from the properties of individual agents alone [87].

At the most general level, a MAS can be described as consisting of *agents* and their *environment*. While most MAS research focuses on software agents, the concept extends equally well to robots, humans, or even hybrid human-agent teams. The environment defines the context in which agents operate, and agent actions are typically mediated through suitable middleware that governs communication, resource sharing, and coordination.

Agents themselves can range from very simple to highly complex. Categories include *passive agents*, or agents without explicit goals (e.g., objects in a simulation such as obstacles or resources); *active agents* with simple objectives (e.g., animals in flocking models, predators and prey in ecological simulations); and *cognitive agents*, which employ more advanced reasoning and calculations. Similarly, agent environments can be virtual, discrete, or continuous. They can also be characterized according to properties such as accessibility (whether complete information about the environment can be obtained), determinism (whether actions lead to predictable effects), dynamics (whether multiple entities can simultaneously alter the environment), discreteness (the number of possible states and actions), episodicity (whether past actions influence future interactions), and dimensionality (the role of spatial characteristics in decision-making).

Within this broad spectrum, agents can be designed with different levels of sophistication. A *reflex agent*, shown in Figure 1.15, interacts with the environment through sensors and actuators, and makes decisions purely based on condition-action rules, without any internal model of the environment. In contrast, a *learning agent*, illustrated in Figure 1.16, incorporates not only sensors and actuators but also a *learning element*, which improves performance based on experience, and a *performance element*, which guides its behavior in alignment with specific objectives. These two paradigms exemplify the diversity of agent designs that can coexist within a MAS.

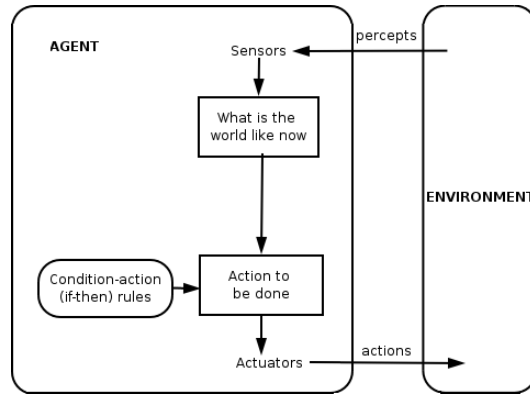


Figure 1.15: Architecture of a reflex agent in a MAS. The agent perceives the environment through sensors and responds with actions through actuators, following simple condition-action rules.

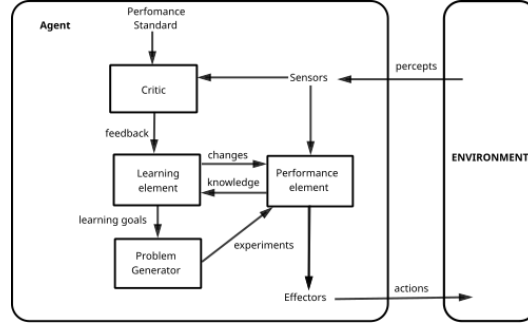


Figure 1.16: Architecture of a learning agent in a MAS. In addition to sensors and actuators, the agent possesses a learning element, which improves its behavior over time, and a performance element, which evaluates and guides its actions.

MASs have been extensively applied across diverse domains, including distributed robotics, economic modeling, resource allocation, social network simulation, and biological systems analysis, owing to their natural ability to represent complex interactions and adaptive behaviors within dynamic environments [27].

Distinction from Agent-Based Models (ABM) While MAS and Agent-Based Models (ABM) share foundational principles, such as decentralized control and interaction-driven dynamics, they differ significantly in their conceptual focus and design goals. ABMs primarily aim at understanding how collective phenomena and global patterns emerge from the local rules and interactions of relatively simple agents. The agents in ABM typically have straightforward, rule-based behaviors, focusing predominantly on collective dynamics rather than on sophisticated individual decision-making[74].

In contrast, MAS emphasize the complexity and autonomy of individual agents, often endowed with intricate decision-making processes, advanced communication protocols, and negotiation mechanisms [100]. MAS agents can be heterogeneous and equipped with capabilities like planning, learning, and reasoning, allowing the simulation of systems where individual behaviors significantly influence overall system dynamics.

MASFENON as a Hybrid MAS-ABM Framework The proposed framework, MASFENON, functions as a hybrid simulation approach by integrating essential elements from both Multi-Agent Systems (MAS) and Agent-Based Modeling (ABM) methodologies, specifically tailored for high-performance computing (HPC) environments. On one side, MASFENON embodies core MAS features by explicitly defining individual agents with sophisticated internal states, decision-making capabilities, and detailed internal topologies. This internal complexity allows each agent to autonomously adapt and react to both internal and external stimuli, thus reflecting the

MAS paradigm, where structured, rule-based inter-agent communication and coordination are paramount.

Simultaneously, MASFENON incorporates crucial aspects from the ABM paradigm by emphasizing the emergent, collective phenomena arising from the interactions among simpler components. In practice, agents in MASFENON behave according to local rules and dynamics, yet the primary modeling objectives often center around capturing and analyzing the emergent patterns and behaviors observable at the system level, such as network diffusion, pattern formation, or regulatory feedback loops.

MASFENON’s unique hybrid approach is particularly advantageous in HPC and distributed computing contexts because it permits clear separation between the agent-level computations (intra-agent dynamics) and the inter-agent communication mechanisms. Agent-level computations are efficiently parallelized using high-performance libraries (e.g., Armadillo and OpenMP), allowing fine-grained agent details and internal states to be modeled without sacrificing computational performance. Inter-agent interactions leverage distributed computing strategies (via MPI), allowing efficient simulation of complex, multi-layered, and large-scale systems. This hybridization thus supports optimized scalability, enabling MASFENON to handle simulations that capture both intricate agent-specific behavior (typical of MAS) and high-level emergent system dynamics (typical of ABM).

Therefore, MASFENON’s hybrid MAS-ABM nature not only provides robust flexibility in terms of modeling granularity but also facilitates computational efficiency and scalability, making it highly suitable for diverse, data-intensive applications in biological, social, and technological domains.

This hybrid nature allows MASFENON to effectively address the challenges in simulating complex, real-world systems such as biological networks, where both sophisticated individual behaviors (e.g., gene expression, regulatory mechanisms, signaling events) and collective dynamics (e.g., tissue differentiation, cellular population behaviors) must be concurrently represented. As such, MASFENON bridges the gap between microscopic agent-level detail and macroscopic emergent system-level properties, providing a robust platform for accurate simulation and analysis of interconnected network phenomena.

1.4.2 Core Characteristics of MAS

Multi-agent systems (MAS) are distinguished by several core characteristics that enable them to effectively model complex, decentralized, and interactive systems. Understanding these characteristics is essential for recognizing the suitability and advantages of MAS approaches in various application domains.

Autonomy One of the defining characteristics of agents within a MAS is autonomy. Each agent operates independently, making decisions based on its internal state and the information available from its interactions with other agents and the environment. Agents are not directly controlled by external entities but rather govern themselves according to predefined rules, goals, or learning mechanisms [100].

Interactivity and Communication Agents within a MAS interact and communicate to achieve individual or collective goals. Interactions may include information exchange, negotiation, cooperation, or competition. These communications facilitate coordination and collective decision-making, influencing overall system behavior and leading to emergent properties [27].

Reactivity and Proactivity Agents exhibit both reactive and proactive behaviors. Reactivity refers to the agent's ability to perceive changes in the environment and respond accordingly, while proactivity implies agents' capability to initiate actions based on internal goals rather than purely external stimuli. This duality allows agents to adapt dynamically, contributing to the robustness and flexibility of the MAS [87].

Sociality and Collaboration Sociality emphasizes the agents' ability to form dynamic, organized groups or networks, facilitating cooperative problem-solving and collective action. Agents collaborate to accomplish tasks that are beyond individual capacities, enhancing the overall efficacy of the system. Mechanisms for collaboration may include task allocation, joint decision-making, and consensus-building processes [74].

Adaptivity and Learning Adaptivity and learning capabilities enable agents to improve their performance over time based on past experiences. Through mechanisms such as reinforcement learning, evolutionary strategies, or supervised learning, agents can adjust their behaviors, refine decision-making processes, and optimize interactions to achieve better outcomes. Adaptivity ensures sustained effectiveness in changing or uncertain environments [22]. These capabilities are also really important for the state of the art today, since most of the multi-agent systems deploy some sort of learning.

Emergence Emergence refers to complex, system-level phenomena that arise from relatively simple interactions among agents without centralized control. This characteristic is pivotal in MAS, as it allows the study and simulation of how micro-level interactions lead to macro-level outcomes, often unexpected or counterintuitive, providing valuable insights into the dynamics of real-world systems [74].

Heterogeneity Heterogeneity in MAS describes the diversity of agent types, roles, behaviors, and interactions within the system. Agents may differ significantly in capabilities, goals, decision-making logic, or interaction patterns. This diversity allows MAS to realistically model systems characterized by varied and specialized roles, mirroring real-world complexity and interdependencies [27].

Understanding these core characteristics provides a comprehensive foundation for leveraging MAS approaches in modeling, simulating, and analyzing complex, decentralized systems across various domains, including biology, economics, robotics, and social science.

1.4.3 Agent Architectures

Agent architecture refers to the internal design and structural components that define an agent’s decision-making processes and behaviors in a multi-agent system (MAS). A robust agent architecture not only determines how an agent interacts with other agents and its environment but also influences the overall efficiency, adaptability, and scalability of the MAS itself.

Agent architectures can generally be categorized into three primary types:

1. **Reactive Architectures:** In this simplest form of architecture, agents respond directly to environmental stimuli without internal state representation or planning. The behavior of reactive agents is characterized by a set of condition-action rules that map observed states directly to actions. Reactive architectures are ideal for environments that are dynamic and unpredictable, where immediate responses are essential. Examples of reactive architectures include Subsumption Architecture and Behavior-Based Robotics.
2. **Deliberative (Cognitive) Architectures:** These architectures involve symbolic reasoning and internal state representations. Deliberative agents incorporate models of their environment, goal structures, and plans for future actions. They typically employ explicit reasoning mechanisms, such as planning, logic, and goal-directed strategies, allowing agents to act based on anticipated future states rather than purely on immediate stimuli. Common deliberative architectures include Belief-Desire-Intention (BDI) architectures, Soar, and ACT-R.
3. **Hybrid Architectures:** Hybrid architectures combine the responsiveness of reactive agents with the advanced reasoning capabilities of deliberative agents. This combination allows for both quick reaction to environmental changes and sophisticated long-term planning and decision-making. Hybrid systems are advantageous in complex, dynamic environments where adaptive behavior is crucial. Notable examples of hybrid architectures include TouringMachines and InteR-RaP.

Components of Agent Architecture While specific implementations vary, most MAS architectures share common components:

- **Perception Module:** Responsible for acquiring and interpreting environmental data through sensors.
- **Reasoning Module:** Contains the agent’s decision-making logic, planning, and goal management systems.
- **Knowledge Base:** Stores information and internal models about the environment, tasks, and other agents.
- **Communication Module:** Enables interaction and information exchange with other agents.
- **Action Module:** Translates decisions into concrete actions performed in the environment.

In the context of the MASFENON framework, the agent architecture adopts a hybrid model. Each agent not only reacts swiftly to changes and interactions at a local level but also engages in higher-level reasoning and decision-making, reflecting the complexity and dynamics of biological and other interconnected networks.

1.4.4 Communication and Coordination

In a Multi-Agent System (MAS), agents rarely operate in isolation. Instead, the hallmark of such systems lies in the capacity of agents to interact with each other in structured ways to achieve local or global objectives. This interaction manifests through two tightly linked aspects: communication, which enables the exchange of information, and coordination, which governs the alignment of actions among agents. Together, these properties form the basis of distributed decision-making and system coherence in MAS environments.

Agent Communication Agent communication refers to the transfer of data, requests, and responses between autonomous agents. This communication can be:

- **Direct (explicit):** Messages are passed using formal languages or protocols (e.g., KQML [31]), typically consisting of speech acts like **inform**, **request**, or **query**. Messages and events can also be sent directly or requested from message queues for an agent or a specific topic (MPI and also reactive programming use this approach)
- **Indirect (implicit):** Agents may communicate via environmental changes (stigmergy), such as in swarm-based or spatially embedded systems, where agents respond to signals embedded in the environment.

Communication protocols are often layered, defining not just syntactic message formats, but also the semantics and pragmatics of interaction. For instance, the FIPA (Foundation for Intelligent Physical Agents) standards

[65] define a robust interaction protocol based on performative acts, allowing for extensible and flexible communication schemes.

Coordination and Cooperation While communication handles the exchange of information, coordination ensures that agents act in a way that avoids conflict and maximizes synergy. Several mechanisms exist to achieve coordination:

- **Centralized coordination:** A single agent (or small subset) orchestrates the plan execution and mediates the behavior of others. This *orchestration*-based approach simplifies management but introduces a potential bottleneck and a single point of failure.
- **Decentralized coordination:** Agents decide autonomously while following shared interaction rules. This resembles *choreography*, where no single agent controls the system; instead, each agent follows a local protocol, and global order emerges from their interactions [69]. Such models are more scalable and robust but require sophisticated negotiation protocols, such as consensus algorithms or contract nets.
- **Market-based or game-theoretic coordination:** Agents use utility functions and competitive strategies to make decisions. Auctions, bargaining, or cooperative game theory can define the negotiation landscape.
- **Emergent coordination:** In swarm systems or biologically inspired MAS, global order arises from local interactions without explicit global rules [10].
- **Event-driven coordination:** Agents respond to events generated within the system or environment, enabling loosely coupled, asynchronous interactions. Event-driven middleware is often employed to propagate state changes efficiently across large agent populations [54].
- **Point-to-point coordination:** In scenarios requiring direct interaction, agents may establish explicit one-to-one communication channels, enabling fine-grained synchronization or targeted cooperation at the cost of scalability.

In scenarios involving shared resources or goals, cooperation becomes essential. Cooperation involves forming coalitions, sharing subgoals, and synchronizing behaviors. Protocols for task allocation (e.g., the Contract Net Protocol) and consensus (e.g., leader election, voting schemes) are commonly used in this context.

Challenges in Communication and Coordination Key challenges include:

- **Scalability:** As the number of agents grows, message complexity and synchronization overhead can become significant.
- **Robustness:** Communication failures or partial observability may

hinder coordination.

- **Heterogeneity:** Agents with different capabilities or objectives may interpret messages differently or conflict during coordination.
- **Latency and asynchrony:** Ensuring consistency and coherence in asynchronous or time-delayed environments is non-trivial.

Relevance to MASFENON In the MASFENON framework, both local and global coordination mechanisms are considered. Agents (which may represent cells, molecules, or abstract entities depending on the simulation layer) communicate via shared states or messaging queues and may adapt their behavior based on neighbor actions, simulated environment cues, or received stimuli. This aligns well with hybrid communication paradigms, balancing the structure of explicit message passing with the flexibility and robustness of indirect coordination.

1.4.5 Organizational Structures in MAS

In a Multi-Agent System (MAS), the organization of agents, how they are grouped, hierarchically structured, or functionally divided, plays a fundamental role in defining the behavior, scalability, adaptability, and performance of the system. The organizational structure governs how agents are arranged and how roles, responsibilities, and relationships are established and evolve over time.

Why Organization Matters While individual agents act based on local perceptions and internal goals, MAS often requires coordination to address system-level tasks. Without some form of structure:

- Communication can become inefficient or overwhelming.
- Resource contention may arise due to poor task delegation.
- Emergent behavior may deviate from desirable system outcomes.

Thus, organizational structures provide constraints and affordances that simplify interactions, enhance modularity, and enable dynamic adaptation.

Types of MAS Organizational Structures Various organizational paradigms have been proposed, each suited to different environments and problem types:

- **Hierarchical:** Agents are arranged in a tree-like structure with clear lines of authority or control. Parent agents delegate tasks to children. This is efficient for command-and-control or centralized planning models but may limit robustness.
- **Flat (Peer-to-Peer):** All agents operate at the same level, interacting with one another directly. Coordination is emergent or handled through negotiation or consensus protocols. While this offers high robustness, it can suffer from communication overhead.

- **Holonic (Nested Hierarchies):** Combines hierarchy with modularity. Agents can form higher-level "holons" (a holon is both a whole and a part) which act as composite agents. This recursive structure enables scalable design and dynamic reorganization.
- **Coalition-based or Role-based:** Agents form temporary or permanent groups based on shared goals or roles. These structures are dynamic and are often used in task-oriented environments or for resource allocation (e.g., contract net).
- **Organizational Metamodels (e.g., MOISE [35]):** Some MAS platforms define explicit metamodels of roles, groups, and norms (e.g., MOISE uses roles, missions, and norms to enforce social rules in agent behavior).

Agent Roles and Responsibilities Organizational structures often define roles that agents can adopt. A role encapsulates expectations about behavior, capabilities, and interaction protocols. Agents may:

- Switch roles dynamically (e.g., from explorer to coordinator).
- Belong to multiple groups or organizations.
- Learn roles through reinforcement or policy transfer.

Formally, if $\mathcal{A}$ is the set of agents and $\mathcal{R}$ the set of roles, we can define an assignment function $f : \mathcal{A} \rightarrow \mathcal{R}^*$ associating agents with one or more roles, constrained by the current organizational state.

Adaptation and Reorganization A key feature of MAS is reorganizability: the ability to alter the organizational structure at runtime in response to environmental changes, internal failures, or system-level optimization goals. This dynamic restructuring may include:

- Reallocating roles.
- Restructuring hierarchies or coalitions.
- Redistributing coordination responsibilities.

Such adaptability is crucial for simulation frameworks like MASFENON, where agents (e.g., cells or submodules of biological systems) may undergo transformations that change their connectivity, role, or context (e.g., transitioning from a quiescent to an active signaling state).

Implications for MASFENON In the MASFENON framework, organizational modeling is an implicit yet critical aspect. Biological analogs such as tissues, cell types, pathways, or signaling cascades can be interpreted as logical organizations of agents (cells, molecules, etc.). These organizations may change based on time, environmental conditions, or emergent behavior, and must therefore be supported by flexible, extensible structures that allow agents to adopt, modify, and change roles or groups and states over time. It is also very important in the concept of differentiation and cell trajectory,

since the change of cell state could also imply the change in cell type as well. This behaviour is partially implemented in MASFENON, but future development can also take into account the adaptation of the type.

1.4.6 Applications of MAS

Multi-Agent Systems (MAS) provide a natural and flexible framework for modeling, analyzing, and simulating complex systems composed of multiple interacting components. Their ability to represent distributed entities with local goals and partial information, as well as their support for communication, coordination, and adaptation, has made MAS an indispensable paradigm across a variety of domains.

Engineering and Robotics In robotics, MAS enables the development of multi-robot systems for exploration, task allocation, and cooperative behavior. Examples include:

- **Swarm robotics**, inspired by biological collectives (e.g., ants, birds), where simple agents cooperate to achieve global objectives such as area coverage or object retrieval.
- **Cooperative autonomous vehicles**, coordinating in traffic systems or search-and-rescue missions.
- **Modular robotics**, where physically reconfigurable units self-organize into complex morphologies or adapt to different terrains.

Smart Systems and Infrastructure MAS are foundational to the design of smart grids, intelligent transportation systems, and sensor networks, where numerous distributed entities (e.g., sensors, vehicles, energy producers/consumers) must:

- Act locally while optimizing global performance.
- Respond to dynamic changes (e.g., failures, demand shifts).
- Coordinate resource usage or load balancing.

In these systems, MAS provides robustness, scalability, and decentralization, core requirements for cyber-physical infrastructures.

Economics and Markets In economic modeling and simulations of markets or negotiation scenarios, MAS allows:

- Modeling autonomous buyers/sellers with private goals and strategies.
- Simulating complex behaviors such as price formation, competition, and market equilibrium.
- Studying emergent macroeconomic phenomena from micro-level interactions.

This has applications in auction theory, supply chain modeling, and policy simulation.

Social Simulation and Collective Behavior MAS are used in sociological and behavioral simulations to model populations, opinion dynamics, or crowd behavior. Agents represent individuals or groups, and their interactions model social influence, cooperation, or conflict.

For example:

- Epidemic spread models, where agents interact in a contact network.
- Evacuation or urban planning simulations using pedestrian agents.
- Modeling cultural evolution or social norm propagation.

Biology and Systems Medicine MAS have become increasingly relevant in computational biology, especially in simulating:

- **Cellular systems**, where agents represent cells interacting via signaling pathways.
- **Tissue growth and morphogenesis**, incorporating spatial constraints and differentiation.
- **Immune response**, where immune cells are modeled as autonomous agents responding to pathogens or vaccines.
- **Drug delivery or metabolic modeling**, especially when simulating local diffusion, resource competition, or molecular interactions.

MAS are particularly suited to modeling biological systems due to:

- Their capacity for local rules and emergent dynamics.
- Their ability to incorporate both discrete and continuous processes.
- Their modular extensibility to heterogeneous agent types (e.g., integrating molecular, cellular, and tissue-level models).

Applications in MASFENON In the context of this research, MASFENON applies MAS principles to simulate interconnected biological systems or topological systems, particularly focusing on agent-based modeling of gene expression, cell signaling, and communication networks. It is specifically designed to:

- Capture spatial and topological relationships between biological agents.
- Allow agents (e.g., cells, gene products) to operate with autonomy while responding to shared stimuli.
- Model processes like signal propagation, expression dynamics, and network adaptation in a distributed, high-performance environment.

Such applications reinforce the suitability of MAS for simulating multi-scale biological processes and the necessity of flexible agent-based abstractions to reflect the modularity, hierarchy, and reactivity of real-world biosystems.

1.4.7 Challenges and Research Directions

While Multi-Agent Systems (MAS) offer a powerful paradigm for modeling distributed intelligence and complex interactions, they also introduce

significant challenges, especially when scaled to realistic or domain-specific applications. These challenges span multiple dimensions, from foundational theoretical issues to engineering concerns in deployment and performance, and they open up several active areas of research.

In the context of HPC and addressing the challenges of large-scale simulations, HPC methodologies for MAS incorporate parallel and distributed computing paradigms such as Message Passing Interface (MPI), Open Multi-Processing (OpenMP), and GPU acceleration. MPI efficiently coordinates computations across distributed systems, enabling extensive parallelism and data distribution. OpenMP provides parallel execution capabilities within single-node architectures, effectively optimizing intra-node performance. Additionally, linear algebra libraries optimized for HPC, such as Armadillo, further enhance computational efficiency. These techniques form the backbone of modern scalable simulations, significantly improving performance in modeling complex systems at scale.

Scalability and Complexity As the number of agents increases, so does the complexity of interactions and the computational cost of simulation or coordination:

- **State space explosion:** Each agent has its own state, and the overall system state becomes exponentially large in the number of agents.
- **Communication bottlenecks:** In communication-intensive scenarios, decentralized coordination requires bandwidth-efficient protocols or approximation techniques.
- **Load balancing:** In parallel or distributed simulations, ensuring even computational workload across resources is non-trivial.

Research direction: Scalable MAS architectures, compression of state/action spaces via abstraction, and decentralized learning algorithms are critical to address these issues. Tools like MASFENON emphasize distributed computation as a key enabler.

Inter-agent Communication and Coordination Ensuring that autonomous agents can exchange meaningful information while preserving autonomy and avoiding unintended interference is a long-standing problem:

- **Semantics of communication:** How agents interpret, trust, and act upon messages is domain-specific and often underdefined.
- **Conflict resolution:** When goals diverge or resources are shared, coordination strategies (e.g., negotiation, voting, auctions) must be employed.
- **Emergent behavior:** Complex, sometimes unpredictable, group-level behavior can arise from local interactions, this is both a strength and a risk.

Research direction: Development of robust communication ontologies,

intention modeling, reputation systems, and emergent behavior prediction remain critical.

Modeling Realistic Agents Creating agents that behave realistically, particularly in simulations of physical, biological, or social systems, is inherently challenging:

- **Behavior modeling:** Agents may require cognitive architectures, stochastic behavior models, or adaptive policies.
- **Learning:** Agents may need to learn from experience, requiring integration with reinforcement learning, imitation learning, or evolutionary algorithms.
- **Multi-scale representation:** In biology or physics, agents might represent components at different levels (molecules, cells, tissues), requiring nested or hierarchical models.

Research direction: Hybrid models integrating machine learning with symbolic or mechanistic modeling are being explored, as are meta-modeling strategies to adapt agent behavior during simulation.

Verification and Validation Ensuring correctness of MAS, particularly in safety-critical or scientific domains, remains a major difficulty:

- **Emergence is hard to predict:** Even with well-understood agents, system-level behavior may be surprising or undesirable.
- **Reproducibility:** Especially in stochastic or learning agents, reproducibility of experiments and simulation outcomes is not trivial.
- **Ground truth scarcity:** In biological modeling, real-world data to validate against may be incomplete, noisy, or missing entirely.

Research direction: Formal verification of MAS (e.g., using model checking), sensitivity analysis, and surrogate modeling for validation are growing areas.

Integration with Other Paradigms Real-world problems often require MAS to interface with other computational paradigms:

- **Hybrid modeling:** Combining MAS with equation-based models (e.g., PDEs for diffusion), logic-based reasoning, or rule systems.
- **Multi-paradigm simulation:** For instance, integrating cellular automata, discrete-event simulations, and MAS in the same framework.
- **Big data and AI:** Incorporating omics data, large-scale logs, or AI predictions into agent behavior requires careful modeling of uncertainty and heterogeneity.

Research direction: Multi-paradigm simulators, data-driven MAS, and new formalisms for hybrid systems are at the forefront, especially for domains like systems biology and socio-environmental modeling.

High-Performance and Parallel Execution For large-scale MAS, especially when applied to biological systems, performance is a key concern:

- **Simulation latency and throughput:** Simulating millions of agents in biologically realistic timeframes is resource-intensive.
- **Concurrency and race conditions:** In parallel environments, preserving causality or message ordering can be non-trivial.
- **Portability and extensibility:** Frameworks must balance performance with developer usability and the ability to adapt to new domains.

Research direction: High-performance computing (HPC) MAS frameworks, including GPU/MPI-based platforms like MASFENON, are being developed to support scalable, parallel, and real-time simulation.

Ethical and Interpretability Concerns As MAS systems are applied in domains like economics, epidemiology, and health, ethical and interpretability concerns arise:

- **Black-box behavior:** It may be difficult to understand why agents or the system behave in a certain way, especially with learned policies.
- **Bias and fairness:** MAS may propagate or amplify biases present in the data or model design.
- **Explainability:** For MAS-based decision support systems, traceability of agent decisions is essential.

Research direction: Tools for explainable MAS, human-in-the-loop simulation, and governance frameworks are needed, especially when MAS output is used for policy or medical recommendation.

1.4.8 MAS Platforms and Frameworks

The landscape of Multi-Agent Systems (MAS) platforms has evolved significantly, especially with the integration of Large Language Models (LLMs), decentralized coordination, and high-performance computing. These platforms can be broadly categorized into four major classes:

General-Purpose MAS Frameworks

- **JADE (Java Agent DEvelopment Framework)**[7]: A mature Java-based platform compliant with FIPA standards, facilitating agent communication, behavior modeling, and lifecycle management.
- **Jason**[11]: Built on AgentSpeak[75], this framework emphasizes logic-based agent programming, focusing on beliefs, desires, and intentions.
- **GAMA (GIS & Agent-based Modeling Architecture)**[90]: Combines geographic information systems with agent-based modeling, offering a domain-specific language (GAML) for complex simulations.

Simulation-Focused MAS Environments

- **Repast**[20]: A toolkit supporting both discrete-event and time-stepped simulations, widely used in social science modeling. As a trivia for this thesis, we wanted to use Repast directly to start the development of MASFENON (C2C-SEPIA at the time), but I noticed that they did not update the framework much, and decided to implement everything by hand with MPI, OpenMP, CUDA, etc.
- **MASON**[55]: A fast, discrete-event simulation library in Java, designed for large-scale agent simulations with a focus on performance.
- **NetLogo**[92]: An accessible platform for modeling complex systems, particularly in educational settings, though less suited for high-performance requirements.

Domain-Specific and High-Performance Frameworks

- **FLAME and FLAME GPU**[19]: Based on the X-Machine formalism, these frameworks support large-scale simulations, with FLAME GPU enabling execution on graphics processing units.
- **BioDynaMo**[13]: Tailored for simulating biological tissue growth and complex multicellular systems, emphasizing parallelism and biological realism.
- **MAFENON**: Developed in the context of this research, MASFENON is a hybrid MAS framework designed for simulating interconnected biological systems, integrating agent-based modeling with HPC capabilities.

LLM-Integrated and Decentralized MAS Frameworks Recent advancements have led to frameworks that support decentralized coordination and integrate large-scale language models:

- **AutoGen**[101]: A Microsoft-backed framework for building autonomous or human-assisted multi-agent AI systems, offering flexible APIs and a no-code GUI (AutoGen Studio).
- **LangGraph**[94]: Part of the LangChain¹ ecosystem, LangGraph enables dynamic agent workflows with support for branching, loops, and asynchronous execution.
- **DynTaskMAS**[104]: A dynamic task graph-driven framework supporting asynchronous and parallel operations in LLM-based MAS systems.
- **AgentNet**[103]: A decentralized framework employing evolutionary coordination for LLM-based MAS, allowing agents to specialize dynamically without centralized control.

¹<https://www.langchain.com/>

Outlook The integration of LLMs and decentralized architectures in MAS frameworks marks a significant shift toward more intelligent and scalable systems. These developments enable modeling of dynamic environments with higher fidelity and computational efficiency. The appropriate choice of framework depends on the application’s requirements regarding scalability, interoperability, domain specificity, and ease of development.

1.5 Synthesis of the Relevant Overviews

The preceding sections have outlined three fundamental domains that provide the conceptual and methodological backbone for this thesis: *sequencing technologies and cell-to-cell communication*, *network theory*, and *multi-agent systems*. Each of these domains addresses a crucial dimension of the problem space that MASFENON is designed to tackle.

Sequencing technologies and the study of cell-to-cell communication reveal the complexity and heterogeneity of biological interactions at the cellular level. They motivate the need for computational frameworks capable of capturing both molecular detail and intercellular dynamics. Network theory, with its well-established metrics and models, provides the structural and mathematical language to represent these interactions, offering a powerful abstraction for studying connectivity, influence, and system-wide properties. Multi-agent systems, finally, contribute the computational paradigm that enables dynamic simulation of autonomous, adaptive entities whose local interactions give rise to emergent global behavior, an essential requirement for modeling complex biological systems beyond static or compartmental assumptions.

Together, these perspectives converge within MASFENON, which integrates biological realism, network formalism, and agent-based adaptivity into a unified simulation framework. This synthesis is not only necessary to represent the intricate nature of biological communication but also positions MASFENON as a distinctive approach in the broader landscape of computational modeling.

The next section builds on this foundation by situating MASFENON in relation to existing frameworks. By comparing methodologies, strengths, and limitations, we highlight both the commonalities and the innovations that distinguish MASFENON as a novel contribution to the study of complex, interconnected systems.

Chapter 2

State of the Art Models

The development of MASFENON was not conceived in isolation, but rather as a direct response to the limitations and challenges encountered in the existing landscape of simulation frameworks, particularly in the domain of complex systems modeling within biology and beyond. As this thesis aims to provide both a conceptual and technical advancement in the modeling of dynamic, multi-agent networks, it is essential to thoroughly contextualize MASFENON within the broader scope of existing tools and methodologies.

This section offers a comparative analysis of state-of-the-art simulation frameworks and modeling paradigms, focusing on their strengths, limitations, and areas of application. Through this analysis, we highlight critical gaps that emerged during the course of this Ph.D. research, gaps which often included a lack of flexibility in modeling heterogeneous agents, poor scalability to large biological networks, and inadequate support for temporally granular and semantically rich simulations.

The comparative study is structured around three key pillars:

- **Review of Existing Simulation Frameworks:** A detailed evaluation of widely used multi-agent simulation tools (such as NetLogo[92], NDLib[79], MASON[55], GAMA[90], and FLAME[19]), emphasizing differences in their underlying assumptions, temporal modeling capabilities, and suitability for simulating real-world networks at scale.
- **Assessment of State-of-the-Art Models:** This includes a discussion on classical epidemiological models (SIR, SEIR, SIS), common in network science and computational biology, and bioinformatics simulation approaches for regulatory and signaling systems, highlighting their structural limitations and computational boundaries.
- **MASFENON’s Comparative Innovation:** We present a focused analysis of the design choices behind MASFENON that differentiate it from the above paradigms. This includes its support for fine-grained temporal resolution, heterogeneous agent modeling, high-performance computing deployment, and adaptability across different scientific do-

mains, features that were either absent or underdeveloped in prior frameworks.

The insights gathered from this comparative process were instrumental during the early phases of this Ph.D., directly guiding the architectural decisions, technical specifications, and biological modeling capabilities implemented in MASFENON. By understanding what current tools could not yet achieve, especially in terms of agent-level biological fidelity, simulation extensibility, and scalability on HPC architectures, this work provides not only a critical reflection on the state of the art, but also a tangible contribution to the field of network simulation in biology and beyond.

2.1 Review of Existing Simulation Frameworks

In this subsection, we examine prominent simulation frameworks relevant to multi-agent systems (MAS), focusing on their scalability, temporal granularity, adaptability, and computational performance. The frameworks reviewed include NetLogo, NDLib, MASON, GAMA[90], and FLAME/FLAME GPU.

NetLogo NetLogo[92] is a widely-used agent-based modeling environment designed for simulating natural and social phenomena. It offers an intuitive interface and a rich library of models, making it accessible for educational purposes and rapid prototyping. However, NetLogo has limitations in scalability and computational performance, which can hinder its application in large-scale simulations.

- **Scalability:** Limited; performance degrades with large agent populations.
- **Temporal Granularity:** Supports discrete time steps; lacks support for continuous or event-driven simulations.
- **Adaptability:** High for simple models; less suited for complex, domain-specific simulations.
- **Computational Performance:** Moderate; not optimized for high-performance computing environments.

NetLogo will be one of the framework that we will consider when doing a comparison for the performance in 4, although the scope of NetLogo is a lot more general and is not aimed at performance itself.

NDLib NDLib[79] is a Python library designed to model and analyze diffusion processes over complex networks. Built on top of NetworkX¹, it provides a suite of diffusion models and supports both static and dynamic network topologies.

- **Scalability:** Moderate; suitable for medium-sized networks.

¹<https://networkx.org>

- **Temporal Granularity:** Supports both discrete and continuous-time models.
- **Adaptability:** High; allows customization of diffusion models and network structures.
- **Computational Performance:** Moderate; performance depends on the underlying NetworkX library.

NDLib will also be one of the framework that we will consider when doing a comparison for the performance in 4.

MASON MASON (Multi-Agent Simulator Of Neighborhoods) [55] is a fast, discrete-event multi-agent simulation library core in Java, designed to support large-scale simulations. It provides a clean separation between model and visualization, facilitating headless batch runs and integration with other systems.

- **Scalability:** High; capable of handling large agent populations efficiently.
- **Temporal Granularity:** Supports discrete-event simulation.
- **Adaptability:** High; flexible architecture allows for diverse modeling needs.
- **Computational Performance:** High; optimized for performance, suitable for complex simulations.

GAMA GAMA (GIS Agent-based Modeling Architecture)[90] is an open-source modeling and simulation environment for building spatially explicit agent-based simulations. It integrates GIS data and supports multi-level modeling, making it suitable for complex spatial simulations.

- **Scalability:** Moderate; performance can be impacted by complex spatial data.
- **Temporal Granularity:** Supports both discrete and continuous time models.
- **Adaptability:** High; designed for domain experts with a dedicated modeling language (GAML).
- **Computational Performance:** Moderate; performance depends on model complexity and spatial data handling.

FLAME and FLAME GPU FLAME (Flexible Large-scale Agent Modelling Environment)[19] and its GPU-accelerated counterpart, FLAME GPU, are frameworks for building agent-based models that can run on high-performance computing systems. They use a formal X-machine model to define agent behavior and interactions.

- **Scalability:** Very High; designed for large-scale simulations on HPC systems.
- **Temporal Granularity:** Supports discrete-event simulation.

- **Adaptability:** Moderate; requires adherence to the X-machine formalism.
- **Computational Performance:** Very High; optimized for parallel execution on CPUs and GPUs.

Table 2.1: Comparison of Classical Simulation Frameworks

Framework	Scalability	Temporal Granularity	Adaptability	Performance
NetLogo	Low	Discrete	High	Moderate
NDLib	Moderate	Discrete/Continuous	High	Moderate
MASON	High	Discrete-Event	High	High
GAMA	Moderate	Discrete/Continuous	High	Moderate
FLAME/ FLAME GPU	Very High	Discrete-Event	Moderate	Very High

Summary Comparison This comparative analysis highlights the strengths and limitations of existing simulation frameworks. While tools like FLAME GPU offer exceptional performance and scalability, they may lack flexibility in model design. Conversely, frameworks like NetLogo and GAMA provide user-friendly environments but may not scale well for large, complex simulations. MASON strikes a balance between performance and adaptability, making it suitable for a wide range of applications.

In the subsequent sections, we will delve deeper into the evaluation of state-of-the-art models and elucidate how MASFENON addresses the identified gaps, offering a flexible and high-performance solution for simulating complex biological systems.

2.2 Evaluation of State-of-the-Art Models

Many simulation studies in biological and social domains rely on established mathematical models that have stood the test of time due to their simplicity, interpretability, and analytical tractability. Among the most widely used are compartmental models, static network simulations, and Boolean or constraint-based systems. While effective in certain use cases, these approaches exhibit several limitations in modeling complex, heterogeneous, and dynamic systems such as those found in multi-cellular biology or network-driven phenomena.

Compartmental Models: SIR, SEIR, SIS Classical epidemiological models, such as the Susceptible-Infectious-Recovered (SIR), Susceptible-Exposed-Infectious-Recovered (SEIR), and Susceptible-Infectious-Susceptible (SIS) frameworks, divide populations into discrete compartments with constant transition probabilities over time. These models assume well-mixed populations and homogeneous transmission rates.

While tractable and analytically solvable in many cases, these models fall short when applied to:

- **Heterogeneous topologies:** where individual-level structure (e.g., network-based transmission) influences propagation dynamics.
- **Stochastic variability:** as real systems often deviate from deterministic predictions due to individual differences and chance events.
- **Behavioral feedback:** which alters dynamics in response to risk, intervention, or communication patterns, something not captured in traditional models.

We will delve further into the comparison with classical compartmental models, as one of the applications of MASFENON was in the context of epidemics. This and more will be seen in chapter 4.

Diffusion and Influence Models In network science, standard models such as the Independent Cascade and Linear Threshold model are frequently used to simulate contagion or diffusion processes over graphs. While these offer more topological realism than compartmental models, they too are constrained by:

- Fixed update rules that are not temporally adaptive.
- Discrete-time steps that miss finer-grained temporal phenomena.
- Limited capacity for multi-agent heterogeneity or learning.

Boolean and Rule-Based Models Boolean network models and rule-based systems are often employed to study gene regulatory networks (GRNs) or signal transduction pathways. Each gene or protein is represented as a binary variable (ON/OFF), with state transitions governed by logical rules.

Though useful for inferring qualitative system behavior, these models:

- Struggle with quantitative precision or dynamic range.
- Cannot model continuous interactions or graded responses.
- Often assume synchronous updates, which may not reflect biological asynchrony.

Constraint-Based Models Constraint-based approaches like Flux Balance Analysis (FBA)[67] have become a cornerstone in metabolic network simulations. They formulate the problem as a linear program over stoichiometric matrices, optimizing fluxes given a biomass or energy objective.

These models, however, are:

- **Static in nature:** unable to simulate temporal dynamics or regulation.
- **Population-level abstractions:** that lack single-cell resolution.
- **Assumption-heavy:** particularly around optimality, which is often biologically implausible.

Summary and Context Despite the considerable contributions of these models to scientific understanding, they tend to fall short in simulating modern data-rich biological systems where:

- Temporal heterogeneity and fine-resolution dynamics are essential.
- Systems are open, adaptive, and exhibit multi-scale interactions.
- Emergent behaviors arise from heterogeneous, interacting agents or sub-systems.

The limitations of these state-of-the-art models motivated the development of MASFENON, a general-purpose, modular, and multi-agent simulation framework designed to overcome these barriers and model dynamic, large-scale systems with fine control over individual behavior, topology, and signal propagation. The next section will delve into this comparative advantage.

2.3 Comparison and Innovation: MASFENON’s Contributions

One of the fundamental objectives of this research was to develop a simulation framework that bridges the gap between biological realism, computational flexibility, and performance at scale. In this regard, MASFENON (Multi-Agent Simulation Framework for Interconnected Networks) introduces a paradigm shift by blending principles from network science, multi-agent systems (MAS), and high-performance computing (HPC) into a cohesive and extensible system. This section offers a structured comparison between MASFENON and the existing tools reviewed earlier, while highlighting its principal innovations and contributions to the field.

Architectural Flexibility and Generality. Unlike narrowly scoped simulation tools that are often tailored to epidemiology (e.g., SIR-type models) or specific social dynamics (e.g., opinion spread), MASFENON operates as a general-purpose simulation backbone. It supports dynamic networks, hybrid interaction rules (deterministic/stochastic), and allows easy integration of domain-specific logic, whether biological (e.g., GRNs, metabolic networks, etc.), ecological, or artificial. MASFENON treats agents as customizable units with full behavioral and state control, unlike frameworks where agent behaviors are often constrained by hard-coded rules or limited scripting layers.

Temporal Granularity and Asynchronous Computation. A major limitation of synchronous timestep-based frameworks (e.g., NetLogo, GAMA) is their coarse temporal resolution. MASFENON introduces temporal decoupling mechanisms and supports fine-grained, asynchronous interactions at the level of individual agents. This is especially critical in biological systems

where reactions, diffusion, and expression events may occur on heterogeneous temporal scales. The engine enables users to model time not just as discrete steps but as event-driven, continuous, or multiresolution, supporting both snapshot analysis and cumulative dynamics.

Scalability and Performance through HPC Integration. The implementation of MASFENON exploits parallelism at multiple levels (agent-level, message passing, and task distribution) using MPI, OpenMP, and optionally GPU offloading for computation-intensive models. Comparative performance evaluations show that MASFENON can handle networks with millions of agents and billions of interactions, maintaining responsiveness and reproducibility even under extensive parameter sweeps. The design explicitly supports simulations on supercomputers and cluster systems, as seen in the experimental phase of this thesis.

Support for Multi-Layer and Interconnected Networks. MASFENON inherently supports multi-layer graphs, representing different dimensions of interaction (e.g., gene regulation, signaling, spatial CCI) via separate but interlinked layers. Agents can exist across layers and operate within heterogeneous network topologies, enabling cross-modal interaction simulations, a critical requirement for realistic modeling of cellular systems and personalized medicine contexts.

Enhanced Observability and Model Embedding. Beyond simulation, MASFENON includes native support for real-time observables, logging hooks, and embedding modules, enabling seamless coupling with machine learning models and Bayesian inference pipelines. These features make the platform well-suited for data-driven simulation workflows, such as fitting gene activity dynamics to scRNA-seq data or adjusting cell behavior parameters based on trajectory inference models.

Plug-in Friendly and Developer-Oriented. From the beginning, MASFENON was designed to serve not just users but also developers and researchers. Its modular C++ core and plugin-friendly architecture allow it to act as a scientific software infrastructure rather than a closed toolkit. Compared to GUI-heavy but black-box tools (e.g., AnyLogic), MASFENON encourages transparency and scientific reproducibility, in line with FAIR computational practices.

Filling the Gaps in the State of the Art. MASFENON specifically addresses the following deficiencies observed in previous frameworks:

- **Lack of support for biological modularity:** Existing tools often fail to represent genes, pathways, and compartments as modular, composable units.
- **Inflexibility in time modeling:** Most do not support mixed or event-based time semantics.
- **Weak scaling on real networks:** Many frameworks struggle with performance and memory usage beyond $\sim 10^4$ agents.
- **Absence of support for hybrid models:** It is rare to find built-in support for combining discrete agent behavior with continuous or rule-based diffusion/reaction dynamics.

Impact and Transferability. By addressing these gaps, MASFENON extends the frontier of what is possible in computational systems biology and complex system modeling. Its general design enables applications beyond the biological domain, such as simulating infrastructure networks, autonomous agents in distributed AI, or synthetic ecologies. Nevertheless, its current implementation and tuning are optimized for cell-to-cell communication, genetic regulatory logic, and biochemical pathways, aligning with the primary objectives of this Ph.D. project.

Table 2.2: Comparative Summary of Simulation Frameworks

Feature	NetLogo[92]	NDLib[79]	GAMA[90]	MASFENON
Programming Paradigm	Domain-specific language (NetLogo)	Python library (networkx-based)	Custom scripting (GAML)	C++, distributed programming and MPI, OpenMP, OpenCL
Temporal Granularity	Discrete timesteps	Discrete time with fixed update rules	Discrete-event	Continuous and discrete-time hybrid (multi-resolution timestepping)
Scalability	Limited to small/medium scale models	Good for medium graphs (thousands of nodes)	High (parallel agents, limited by GUI overhead)	HPC-ready, scales to millions of agents and high-throughput environments
Spatial Modeling	Basic 2D grid space	Graph-based only	Advanced GIS, continuous and grid spaces	Continuous and graph-based, extendable to spatial simulation using graph embeddings
Agent Complexity	Simple rule-based agents	Nodes with fixed update functions	Complex behavior possible, coded in GAML	Network based, fully programmable agents rules, message-passing enabled, embedded internal states and behavior trees
Support for Heterogeneous Agents	Limited	Partial	Full	Full, with adaptive behaviors, different structure and dynamic role assignment
Bioinformatics Relevance	Minimal	Supports epidemiological processes	Some ecological modeling	Focused on modeling biological cell interactions, gene regulation, signaling and custom bio-logic
Extensibility	Limited to NetLogo capabilities	Extendable via Python code	Highly extensible but steep learning curve	Highly extensible with custom modules and backends, plug-in support planned
Parallelism/Performance	No inherent parallelism	No HPC support	Partial (multi-core)	Native support for hybrid parallelism (MPI, OpenMP, GPGPU via OpenCL or CUDA)

Chapter 3

The MASFENON Framework

To address the challenges seen in the previous chapters, we developed MASFENON (Multi-Agent Adaptive Simulation Framework for Evolution in Networks of Networks), a general-purpose simulation environment for analyzing dynamic processes occurring *within* and *between* networked agents. The framework integrates principles of *propagation*, *dissipation*, and *conservation* to model complex interactions at multiple scales, while maintaining scalability and computational efficiency for HPC deployment.

At its core, MASFENON adopts a dual propagation mechanism, **intra-propagation** and **inter-propagation**, to simulate both internal dynamics of agents and their external communication over time. Each agent is represented as a network whose nodes evolve according to local topological interactions, while inter-agent communication is modeled through a temporal multi-graph of contact events. This structure allows the framework to reproduce real-world complexity, including variable interaction intensities, dynamic topologies, and heterogeneous communication modes.

Designed with modularity and extensibility in mind, MASFENON combines concepts from multi-agent systems, graph theory, and chaos theory to enable flexible simulation of heterogeneous, temporally resolved systems. Depending on the domain and level of granularity, agents may share a common network topology or maintain unique structures, with communication links dynamically formed or dissolved across time (see Fig. 3.1 and Fig. 3.2).

While the internal topology of agents remains static throughout simulations in the current implementation, future developments (see Chapter 5 and Chapter 6.4) will introduce intra-temporal topological variability. This enhancement will allow MASFENON to model systems such as dynamic regulatory rewiring or temporally plastic social structures. The resulting system is thus capable of reflecting real-world complexities, such as periodic phenomena, synchronization, cascading failures, or feedback regulation, with high temporal resolution.

As discussed previously in Chapter 1, the development of MASFENON

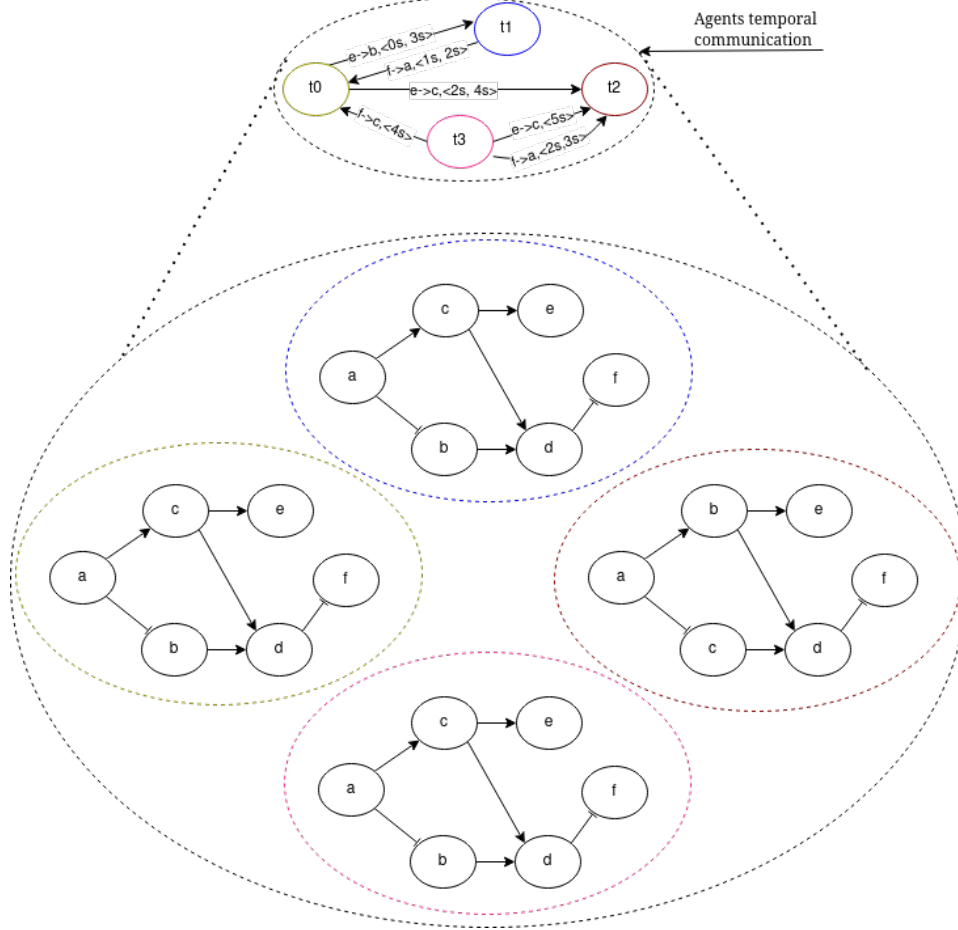


Figure 3.1: Representation of a use case of the MASFENON simulator. Four agents are defined $V(Agents) = t_0, t_1, t_2, t_3$ and for each agent, there is an underlying network that regulates the behavior and the state of the agent. In this case, all the networks of the agents are the same as $G_{Ag} = (\{a, b, c, d, e, f\}, \{(a, b, -1), (a, c, 1), (c, e, 1), (c, d, 1), (b, d, 1), (d, f, -1)\})$ where the edges also have a weight associated with them (which is depicted by the edge arrow type). Singular states are different among the agents. The edges set of the Agents network is $E(Agents) = \{(t_0, t_1, (e, b, \langle 0s, 3s \rangle)), (t_1, t_0, (f, a, \langle 1s, 2s \rangle)), (t_0, t_2, (e, c, \langle 2s, 4s \rangle)), (t_3, t_0, (f, c, \langle 4s \rangle)), (t_3, t_2, (e, c, \langle 5s \rangle)), (t_3, t_2, (f, a, \langle 2s, 3s \rangle))\}$, with the following set of attributes $D \subseteq G_{Ag_1} \times G_{Ag_2} \times \mathcal{C}$. Therefore D will contain the following type of tuples $(v_i, v_j, \langle time1, \dots, timek \rangle)$ where the v_i is the node of the source agent network, v_j is the node of the target agent network, and $\langle time1, \dots, timek \rangle$ is the set of contact times

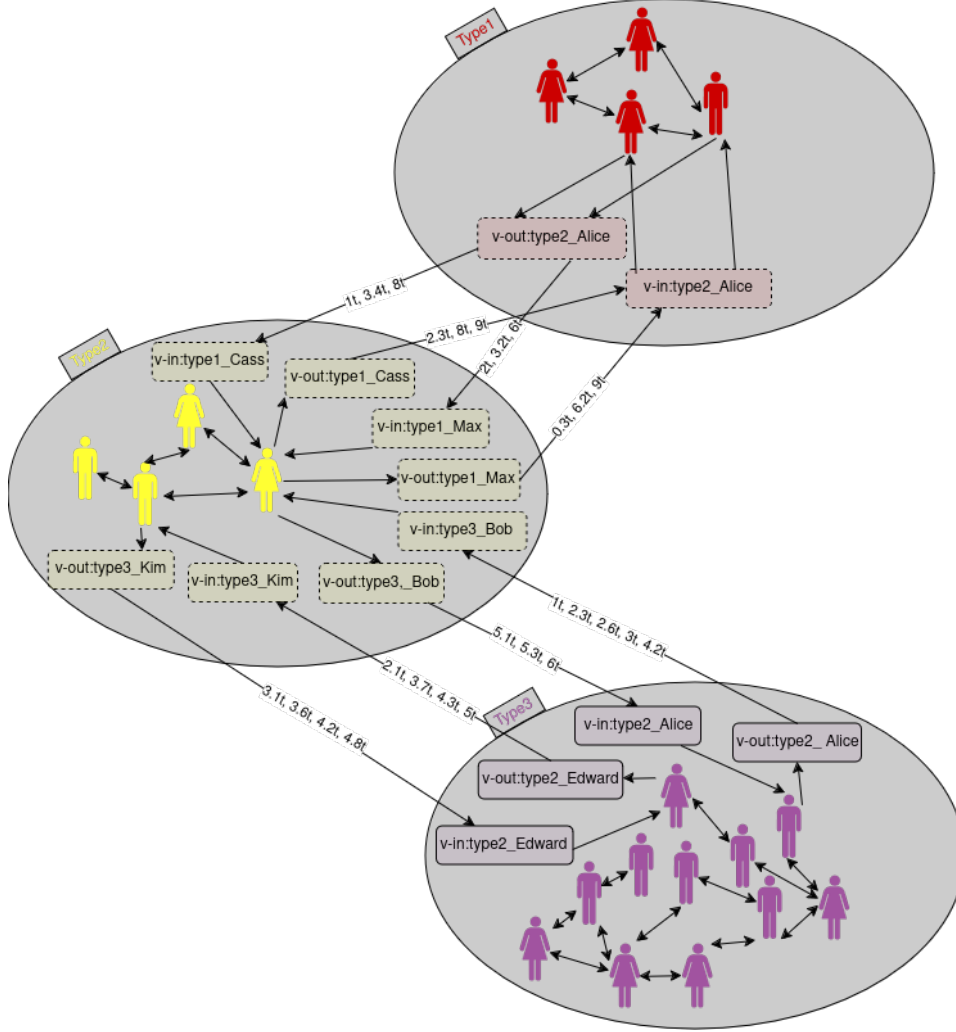


Figure 3.2: MASFENON agent inter-layer communication. Virtual outputs in the source agent/network are linked to virtual inputs in the target agent/network. Communication between different agents occurs through these connections, utilizing the mechanisms described in Section 3.1.5 and in the appendix . The exchange of information is managed by the contact times associated with each virtual node (virtual output, virtual input, and the temporal edge connected to a set of contact times).

was initially rooted in biological modeling and motivated by applications in cellular communication, particularly in the domain of cell-to-cell communication (CCC) and cell-to-cell interaction (CCI), but the framework is designed to be adaptable to any domain that requires the simulation of interactions across structurally heterogeneous and temporally dynamic systems. Whether in epidemiology, social science, or infrastructure modeling, MASFENON provides a modular and extensible framework suitable for interdisciplinary research. Early iterations of the framework (e.g., C2C-SEPIA) were motivated by the challenges encountered during my master’s thesis on single-cell RNA sequencing and precision medicine. That work explored how expression profiles and pathway dynamics could be used to model the temporal evolution of cellular states and their mutual influence within tissue systems. The resulting insights directly informed the dual propagation architecture of MASFENON, which now generalizes these principles across arbitrary networked agents and communication schemas.

Thus, sequencing technologies, notably single-cell RNA-seq, played a foundational role in shaping the framework’s evolution. They emphasized the need for temporally fine-grained, agent-based simulations that account for heterogeneous internal structures and sparse, asynchronous inter-agent signaling. These same requirements continue to guide the design of MASFENON and demonstrate its relevance for applications in systems biology and beyond.

While MASFENON was inspired by biological systems, it was explicitly designed as a domain-agnostic simulation engine. Importantly, the decision to implement MASFENON as an adaptable and generalized framework, rather than a strictly domain-specific tool, stems from the recognition that complex network dynamics share fundamental properties across numerous fields, including biology, epidemiology, social sciences, and economics. This generality reflects the recognition that complex network dynamics exhibit fundamental similarities across disciplines, including epidemiology, infrastructure modeling, economic systems, and information propagation. By supporting flexible architecture, adjustable time modeling, and graph-based propagation dynamics, MASFENON enables researchers to tailor simulations to the structure and behavior of their target systems.

By ensuring adaptability, MASFENON not only meets current simulation demands but also accommodates evolving future requirements and diverse scientific applications. This flexible approach enables researchers and practitioners across multiple domains to apply and customize the framework according to their specific data characteristics, simulation scales, and temporal resolutions. This decision to avoid domain-specific hardcoding also provides practical advantages:

- **Broader dissemination and adoption:** The framework attracts users from multiple scientific communities, increasing collaboration and use-case diversity. Having a wider spectrum of applications also makes

the project appealing to a wider audience of people from different fields of research, and who want to simulate a system based on some kind of assumption or data.

- **Code quality and extensibility:** Open and modular codebases promote contribution, bug detection, and feature evolution over time. The code will be available to a wider audience, finding errors to address or features to implement will be a lot easier since different people will use the framework for different use cases
- **Improved modeling quality:** Customization allows models to match real-world dynamics and incorporate domain-specific assumptions with precision more closely. Getting more features to implement and more ways to do it can bring better and closer to reality results, since the possibility of customization is one of the most flexible ways of having better results.

Optimized for high-performance computing environments, MASFENON leverages advanced distributed computing methodologies, including MPI for inter-agent communication, alongside Armadillo, OpenMP and CUDA for efficient intra-layer computation. These capabilities enable the framework to handle extensive computational demands and ensure robust performance across simulations of vastly different scales.

MASFENON’s implementation leverages high-performance computing (HPC) technologies to meet the intensive computational demands of simulating large, complex systems. The core framework modules are written in **C++** and employ the Message Passing Interface (MPI) standard for distributed computing. Matrix and linear algebra operations are parallelized using **OpenMP**, CUDA and the Armadillo library, while reactive and distributed programming and software design strategies support agent-based simulation logic. This HPC-oriented design enables the framework to scale efficiently across multiple nodes and cores, offering robustness and adaptability for large-scale simulations.

In the following sections, we will elaborate on MASFENON’s conceptual foundations, implementation specifics, computational complexities, and performance analyses, providing a comprehensive view of its architecture and demonstrating its applicability to various complex, temporally dynamic systems.

The accompanying appendix at A.3 provides a full mathematical treatment of the model’s propagation dynamics, correction mechanisms, and stability considerations.

Throughout this chapter and in the appendix, we use the following notation:

- $G_{Ag} = (V(G_{Ag}), E(G_{Ag}))$ is a graph associated to agent $\mathbf{Ag} \in V(Agents)$, representing a type of network, where $V(G_{Ag})$ is the set of nodes of graph G_{Ag} , and $E(G_{Ag})$ is its set of edges. A graph represents a collection of nodes and edges. Each node describes an entity, while an

edge indicates a relationship or interaction between two nodes. We associate a graph with each agent.

- $\mathcal{G}_{Ag} = (V(\mathcal{G}_{Ag}), E(\mathcal{G}_{Ag}))$ is the augmented graph, where $V(\mathcal{G}_{Ag}) = \{v_1, v_2, \dots, v_{dim(Ag)}\}$ are the nodes, $dim(Ag) = |V(\mathcal{G}_{Ag})|$ is the number of nodes associated to an agent, both normal and virtual nodes of the augmented graph $\mathcal{G}_{Ag}$, these nodes are defined as the union of the virtual nodes and the nodes of the normal graph $V(\mathcal{G}_{Ag}) = V(G_{Ag}) \cup V_{in}(Ag) \cup V_{out}(Ag)$. $E(\mathcal{G}_{Ag})$ are the edges of the augmented graph $\mathcal{G}_{Ag}$, denoting the connections between two nodes, defined as the union of virtual edges between communities $E(\mathcal{G}_{Ag}) = E(G_{Ag}) \cup E_{in}(Ag) \cup E_{out}(Ag)$. This augmented graph is obtained with the process of **graph augmentation** (see 3.1.2) to represent the interaction between different components. These added nodes will be called virtual inputs and virtual outputs.
- $V_{in}(Ag) = \{vin_1, vin_2, \dots, vin_q\}$ is the set of virtual nodes representing input information from outside of the network associated with agent Ag . This information can come from components of other agents or from a whole agent, depending on the use case and the granularity defined.
- $V_{out}(Ag) = \{vout_1, vout_2, \dots, vout_z\}$ is the set of virtual nodes representing output information outside of the network associated with agent Ag . This information can go to components of other agents or to the agent itself, like before, depending on the use case and the granularity defined.
- $E_{in}(Ag) = \{(vin_i, v_j) | vin_i \in V_{in}(Ag), v_j \in V(G_{Ag})\}$ is the set of edges with virtual input as source nodes and nodes in the graph G_{Ag} as target nodes. These edges represent the interaction of the agents and their components with incoming information.
- $E_{out}(Ag) = \{(v_j, vout_i) | vout_i \in V_{out}(Ag), v_j \in V(G_{Ag})\}$ is the set of edges with virtual output as target nodes and nodes in the graph G_{Ag} as source nodes. These edges represent the interaction of the agents and their components with outgoing information.
- $\vec{v}^{(n)}(\mathcal{G}_{Ag}) \in \mathbb{R}^{dim(Ag)}$: is the vector of node values at iteration n associated to graph $\mathcal{G}_{Ag}$. Since we associate an infection value to every node in the augmented graph, this vector denotes the infection values of the people (or the buffered propagated infection waiting for the inter-network propagation).
- $\vec{v}_i^{(n)}(\mathcal{G}_{Ag}) \in \mathbb{R}$: is the i -th element of $\vec{v}^{(n)}$, which is associated to the i -th node of $V(\mathcal{G}_{Ag})$. Vector of node values at iteration n , representing perturbation values.
- $U : V(\mathcal{G}) \times \mathcal{G} \rightarrow X, X \subseteq V(\mathcal{G})$ is the function that returns the set of predecessors (upstream) of v_i in the augmented graph $\mathcal{G}$, this set is defined as $U(v, \mathcal{G}) = \{u \in V(\mathcal{G}) | (u, v) \in E(\mathcal{G})\}$.

- $D : V(\mathcal{G}) \times \mathcal{G} \rightarrow X, X \subseteq V(\mathcal{G})$ is the function that returns the set of successors (downstream) of v_u in the augmented graph $\mathcal{G}$, this set of successors (downstream) is defined as $U(u, \mathcal{G}) = \{v \in V(\mathcal{G}) | (u, v) \in E(\mathcal{G})\}$.
- $w_{\mathcal{G}_{Ag}} : V(\mathcal{G}_{Ag}) \times V(\mathcal{G}_{Ag}) \rightarrow \mathbb{R}$ is a weight function associated with each edge $(v_u, v_d) \in E(\mathcal{G}_{Ag})$.
- $\bar{w}_{\mathcal{G}_{Ag}} : V(\mathcal{G}_{Ag}) \times V(\mathcal{G}_{Ag}) \rightarrow [-1, 1]$ is a normalized weight function, associated with each graph edge $(v_u, v_i) \in E(\mathcal{G}_{Ag})$, defined in equation 3.1.
- $\bar{W}(\mathcal{G}_{Ag}) \in \mathcal{M}_{dim(Ag) \times dim(Ag)}(\mathbb{R})$ is the normalized adjacency matrix associated with augmented graph $\mathcal{G}_{Ag}$, derived from the individual normalized weights associated to each edge seen in Eq 3.1 . $\bar{W}(\mathcal{G}_{Ag})^T$ is the transpose of this matrix.
- Agents = $(V(\text{Agents}), E(\text{Agents}))$: Temporal multi-graph of agent interactions between each other, where $V(\text{Agents}) = \{Ag_0, Ag_1, \dots, Ag_s\}$ is the agents/nodes of the graph (associated with an underlying network as said before), and $E(\text{Agents}) = \{(v_i, v_j, d) | v_i, v_j \in V(\text{Agents}), d \in D\}$ is the set of edges among agents, where D is the set of attributes (including contact times) associated to an edge. It should be noted that different nodes can have more than one edge between them; the only constraint is that the attributes of edges between two nodes need to be different. The meaning of these agents changes based on the context of the use case and the modeling that is done. In communities context (epidemics or social networks) Agents represent different communities, while in the context of scRNA-seq, the agents represents the single cell or the single cell type. Interactions between two agents are interactions between two members of two different communities, and these interactions have a list of contact times that will be used to propagate information(infection values in the case of epidemics) at specific times.

3.1 Concept and Design

The conceptual foundation of MASFENON is grounded in a detailed representation of multi-agent interactions in interconnected networks, coupled with a robust temporal network structure. The design explicitly considers the diverse and dynamic nature of interactions that occur both within individual networks and between distinct interconnected networks. The framework was born in the context of cellular dynamics and communication, and it was first called C2C-SEPIA(or SC-Phensim, since it was a rework on some of the concepts seen in [1]). In the context of cellular dynamics, single cells have an internal structure and behavior that can be influenced by communicating with other cells, and their internal state changes through time, both for

internal and external dynamics. This approach encodes information about signaling pathways, networks, and cell expression in scRNA-seq. It creates embeddings, which encode topological information into the node values for each node in the network.

The MASFENON framework is a novel methodology for modeling complex systems, using a multi-agent approach and principles of chaos theory to simulate and analyze dynamic processes in interconnected networks. MASFENON relies on an algorithm that integrates a two-fold communication channel, within individual agents and between them. Every agent has an underlying network topology. In our context, networks have a hierarchical structure in which node connections can happen at different resolution levels.

Additionally, the MASFENON framework is conceived as a flexible and adaptable framework that could be tuned to model the behavior of a real system, this is done by estimating the best parameters that will emulate the real data observations. In this section we will see how the MASFENON framework is characterized and see the mathematical foundations of the framework, to see some in depth analysis of the mathematical characterization of the framework, refer to the appendix A.3

3.1.1 Defining Multi-Agent Interactions in Interconnected Networks

In MASFENON, each *agent* is an autonomous entity equipped with:

- its own **directed attributed graph** (i.e., unique or shared internal topology and node/edge attributes)
- **internal dynamics** governed by propagation, dissipation, and conservation functions. Along some other concepts like saturation, that changes the behavior of the agent and put some constraint on the single agent.

This setup aligns with standard **Multi-Agent System (MAS)** definitions, systems composed of multiple intelligent agents coordinating within a shared environment [28].

In parallel, **Agent-Based Modeling (ABM)** focuses on bottom-up emergence, where agents obey simple rules, and their interactions generate complex, large-scale patterns [23, 57]. MASFENON integrates both paradigms: agents follow rule-based updates (ABM-style) while exchanging structured information across agents (MAS-style) within some external environment conditions and changes (defined by some functions and with the temporal characterisation of inter-agent communication).

- Each agent Ag is modeled by a graph $\mathcal{G}_{Ag}$, which may include virtual input/output nodes for inter-network messaging.
- Agents can **share or own unique topologies**, supporting homogeneous scenarios (shared structure) or heterogeneous ones.

- Internal and virtual connections enable complex inter- and intra-agent behaviors.

Agents evolve internally using an update rule that we will see in 3.1.5, and the update rule is defined in the Eq. 3.8

This formulation represents ABM-style processing: agents autonomously update their internal state based on local rules and topology.

Communication among agents occurs through virtual input/output nodes in the internal topology of the single agent and its interactions with the other agents, with an explicit two-layered approach:

1. **Intra-propagation** runs first for all agents, iterating N_{intra} times.
2. Then, during each **inter-propagation**, the state of each source agent's virtual output node v_y is transferred to the corresponding virtual input node v_x in the target agent, and is also modified in case of certain use cases where the environment where the agents interact could also have some sort of influence in these interactions (also coded with a function). This mimics MAS-style information exchange, similar to message passing in distributed multi-agent frameworks. It also allows the system to model not only physical interaction events, but also abstract or latent exchanges such as influence or regulatory control.

The higher-level inter-agent network is formally defined as a **temporal multi-graph** $\text{Agents} = (V(\text{Agents}), E(\text{Agents}))$, where $V(\text{Agents})$ is the set of all agents, $E(\text{Agents})$ is the set of edges tagged with contact times, forming a dynamic, time-indexed interaction graph.

This captures both structural and temporal dimensions of agent communication.

All of this characterisation is really important for some reasons:

- **Decoupling internal and external processes** supports modularity: internal decisions and external messaging proceed independently but can be tuned jointly.
- The approach combines both ABM's focus on emergent behavior through rule-based agent dynamics, and MAS's structured communication and coordination mechanisms.
- Enables modeling of **heterogeneous agent types**, shared or disparate structural complexities, and rich forms of interaction, ideal for simulations in systems biology, epidemiology, social networks, and infrastructure.

For a full theoretical background and references on MAS and ABM, see Chapter 1, and in particular Section 1.4. This section emphasizes how MAS-FENON inherits and extends core principles from both paradigms for flexible, scalable, multi-layered network simulation. We will see in depth how the framework is defined and what are the capabilities in the following sections, while we will focus on some use-cases and experimental results in Chapter 4 and Chapter 6.

Future development will also work on expanding the capabilities of the

framework environment by considering more complex structures of agents and agent communication, such as implementing **Temporal Hypergraphs**. This and more will be explored in Chapter 5.

3.1.2 Graph Augmentation and Structural Modeling

A critical structural feature of MASFENON is the augmentation of each agent's network graph to enable flexible intra- and inter-agent dynamics. Rather than operating on raw node-level graphs or integrating the communication between different agents in a less standardized way, such as having some sort of passing of the information of the agents and modifying the state of the agent with this information (modifying the topology or the values of the network associated with the agent itself), MASFENON transforms each agent's network into an *augmented graph*, denoted $\mathcal{G}_{Ag}$, which includes both standard network nodes and dedicated **virtual nodes** for communication. This choice enable both a standardized way of encoding the communication between the agents, very important in the context of high performance systems that rely on standardized implementation, easy to develop, maintain and update, and it also enable the analysis of the communication dynamics by observing the behaviour of these virtual nodes and their temporal evolution, meaning that spectral decomposition and analysis can be easily deployed by taking into account only these virtual nodes.

Formally, each agent $Ag \in V(\text{Agents})$ has an underlying graph:

$$G_{Ag} = (V(G_{Ag}), E(G_{Ag}))$$

which is extended to:

$$\mathcal{G}_{Ag} = (V(\mathcal{G}_{Ag}), E(\mathcal{G}_{Ag}))$$

where:

$$V(\mathcal{G}_{Ag}) = V(G_{Ag}) \cup V_{\text{in}}(Ag) \cup V_{\text{out}}(Ag)$$

$$E(\mathcal{G}_{Ag}) = E(G_{Ag}) \cup E_{\text{in}}(Ag) \cup E_{\text{out}}(Ag)$$

Here, $V_{\text{in}}(Ag)$ and $V_{\text{out}}(Ag)$ are sets of virtual input and output nodes, used respectively to receive and send perturbation values during inter-agent propagation. These nodes are uniquely associated with cross-agent interaction endpoints and do not participate in the agent's internal topology outside of propagation events. $E_{\text{in}}(Ag)$ and $E_{\text{out}}(Ag)$ are respectively the virtual input edges (connecting the virtual input nodes to the nodes in G_{Ag}), and the virtual output edges (connecting the nodes in G_{Ag} to the virtual output nodes).

The virtual nodes $V_{\text{in}}(Ag)$ (virtual inputs) and $V_{\text{out}}(Ag)$ (virtual outputs) retain information before sending it to the target type. Since these virtual nodes encode the information of the outside environment, they can

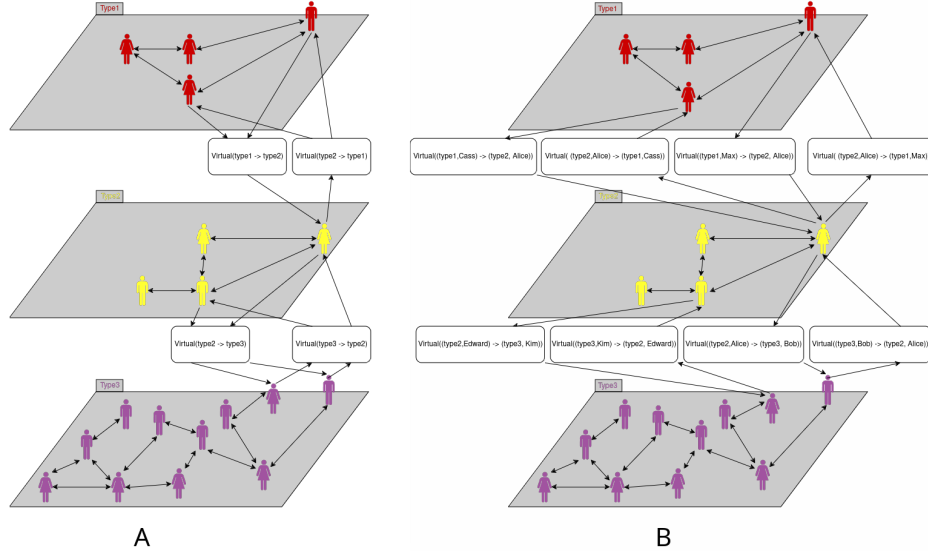


Figure 3.3: Granularity of virtual nodes. In Figure A, **Coarser** granularity is depicted. In this scenario, virtual nodes between two agents are associated with the ordered pair of agents. In the picture this is represented by two virtual nodes: one for the interactions from *Type1* $\rightarrow$ *Type2* and another for the interactions from *Type2* $\rightarrow$ *Type1*; In Figure B, a **finer** granularity is depicted. Virtual nodes between two agents are associated with ordered quadruplets. These quadruplets represent interactions between components of the two different networks associated to the agents.

be associated and deconstructed with different levels of **granularity**. In the MASFENON framework, we have the two following types of granularity:

- **Coarser granularity:** At a coarser granularity level, virtual nodes are associated with relations between two agents. Formally, let (Ag_i, Ag_j) denote a relation between agents $Ag_i, Ag_j \in V(Agents)$. These virtual nodes encapsulate the overall interaction or relationship between the two agents collecting information coming from the nodes within the layer.
- **Finer granularity:** At a finer granularity level, virtual nodes provide a more detailed representation of interactions between individual nodes of different agents. Formally, let (Ag_i, n_k, Ag_j, n_l) be a specific interaction between node $n_k \in G_{Ag_i}$ of agent $Ag_i \in V(Agents)$ and node $n_l \in G_{Ag_j}$ of agent $Ag_j \in V(Agents)$ (see Fig 3.3 B) .

Purpose and Role of Virtual Nodes Virtual nodes provide some key modeling advantages:

- They decouple internal agent dynamics from external influence, enabling modular design.
- They allow for asynchronous communication: virtual output nodes

accumulate internal values during intra-propagation, which are then transferred to virtual inputs in target agents during inter-propagation steps.

- They make inter-agent communication more modular and, consequently, more extensible with things like processing the information that is passed from an agent to another agent, like filtering, changing or combining different outputs to create the inputs. This will have been one of the biggest advantages during production, since new features regarding the communication among agents have been easily implemented thanks to this modeling strategy.
- This structure also facilitates testing, since the behaviour of the intra-propagation is tested in a different setting than inter-agent communication, and the testing itself becomes more easy to implement.

This structure preserves a clean separation between **endogenous** agent evolution and **exogenous** influence, allowing simulation designers to manipulate both processes independently while retaining a consistent graph representation.

Edge Types and Directionality Three categories of edges define connectivity in $\mathcal{G}_{Ag}$:

- **Internal edges** $E(G_{Ag})$: the intrinsic topology of the agent (e.g., pathway interactions or organizational flows).
- **Input edges** $E_{in}(Ag)$: from input virtual nodes to normal nodes.
- **Output edges** $E_{out}(Ag)$: from normal nodes to output virtual nodes.

All edges are weighted and directed, allowing for modeling asymmetric influence and supporting precise computation of propagation using adjacency matrices. The weight of an edge connecting a virtual input to a node in a specific agent is the same as the weight of an edge connecting a virtual output to a node in another agent (the source and target agent).

More formally, given two agents Ag_{source} and Ag_{target} , let $v_{out} \in V_{out}(Ag_{source})$ be a virtual output node in the source agent's augmented graph $\mathcal{G}_{Ag_{source}}$, and let $v_{in} \in V_{in}(Ag_{target})$ be its corresponding virtual input node in the target agent's augmented graph $\mathcal{G}_{Ag_{target}}$. For any normal node $v_j \in V(G_{Ag_{source}})$ and $v_k \in V(G_{Ag_{target}})$, the weight assignments satisfy the following condition:

$$w_{\mathcal{G}_{Ag_{target}}}(v_{in}, v_k) = w_{\mathcal{G}_{Ag_{source}}}(v_j, v_{out})$$

where edges $(v_j, v_{out}) \in E_{out}(Ag_{source})$ and $(v_{in}, v_k) \in E_{in}(Ag_{target})$ represent complementary communication paths between the agents.

This constraint explicitly ensures symmetry in inter-agent communication, where the outgoing influence from the source agent's network is directly mapped onto incoming influence in the target agent's network, preserving the meaning and intensity of the interaction.

Normalized Adjacency and Matrix View The augmented graph’s adjacency matrix $W(\mathcal{G}_{Ag})$ encodes the raw directional edge weights of all connections, including internal, input, and output links. To ensure that propagation distributes influence proportionally, the matrix is row-normalized to yield $\bar{W}(\mathcal{G}_{Ag})$, where each entry is computed as:

$$\bar{w}_{\mathcal{G}_{Ag}}(v_u, v_i) = \begin{cases} \frac{w_{\mathcal{G}_{Ag}}(v_u, v_i)}{\sum_{k \in D(v_u, \mathcal{G}_{Ag})} |w_{\mathcal{G}_{Ag}}(v_u, k)|} & \text{if } \sum_{k \in D(v_u, \mathcal{G}_{Ag})} |w_{\mathcal{G}_{Ag}}(v_u, k)| \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

Here, $D(v_u, \mathcal{G}_{Ag})$ denotes the set of downstream neighbors (successors) of node v_u in the augmented graph $\mathcal{G}_{Ag}$, associated with agent $Ag \in V(\text{Agents})$. The weight normalization procedure produces a value $\bar{w}_{\mathcal{G}_{Ag}}(v_u, v_i)$ defined as the ratio of the edge weight to the sum of absolute weights on outgoing edges from v_u . This normalization guarantees that the total outgoing influence from each node is bounded and comparable, a key requirement for stability in iterative propagation models. The matrix $\bar{W}(\mathcal{G}_{Ag})$ is then used in all core modeling functions (see Eq. 3.2).

This transformation is not a strict mathematical normalization in the classical sense, but rather a rescaling or **range adjustment** operation. The total absolute influence exerted by a node is brought to a unit scale while preserving the original sign and relative magnitude of the interactions.

Crucially, the **sign of each edge weight is preserved** in the normalization, meaning negative relationships remain negative in the resulting normalized matrix. This property ensures that **semantic meaning encoded in the original network**, such as inhibitory vs. excitatory influences or antagonistic vs. synergistic relationships, is maintained throughout the simulation. Another version of this normalization can be seen in the A.3, where the normalized weight is directly set to 0 if the weight of the edge is 0 from the start (the resulting normalized weight is the same, but the formulation is different).

The output of the normalization lies within a fixed codomain of $[-1, 1]$. More precisely:

- If all outgoing weights from a node are positive, the range becomes $[0, 1]$.
- If all are negative, the range becomes $[-1, 0]$.
- If the outgoing edges contain both positive and negative weights, the resulting normalized values fall within $\left[\frac{\min w}{\sum |w|}, \frac{\max w}{\sum |w|} \right]$ where $\sum |w|$ denotes the sum of absolute outgoing weights.

This construction guarantees that each row of the normalized adjacency matrix $\bar{W}(\mathcal{G}_{Ag})$ maintains a unit influence span, which is essential for **numerical stability**, **convergence** and **interpretability** in the iterative

propagation model, especially with the combination of the dissipation and conservation dynamics.

Parameter Assignment Each virtual node and augmented edge can be assigned user-defined parameters (weights, damping coefficients, conservation scalars), allowing simulation of heterogeneous or hierarchical systems. The flexibility of graph augmentation makes MASFENON well-suited to applications requiring directional and layered information flow. The virtual nodes and virtual edges are practically customizable in the same way the normal nodes and edges are, meaning that propagation dynamics can be tuned to have very specific behavior, based upon the needs and the use case of the framework.

For a formal treatment of the augmented graph construction and its role in propagation, see Appendix A.3.

3.1.3 Temporal Network Structure

A key innovation in MASFENON is its ability to represent and simulate temporally dynamic interactions accurately. Traditional network models often assume static topologies or fixed interactions; however, MASFENON explicitly incorporates temporal resolution into its fundamental architecture. Interactions between agents occur at discrete, customizable time intervals, enabling precise modeling of temporal dynamics such as transient phenomena, periodic interactions, and evolving network topologies. This temporal granularity allows researchers to simulate scenarios closer to real-world conditions, significantly enhancing the predictive capability and reliability of simulations.

More formally, the framework constructs a temporal multi-graph of agent interactions:

$$Agents = (V(Agents), E(Agents))$$

Here, $V(Agents) = \{Ag_0, Ag_1, \dots, Ag_s\}$ are the agents, and $E(Agents) = \{e_i | e_i \in V(Agents) \times V(Agents) \times D\}$ are the interactions, with D representing attributes like contact times.

Agent types can be defined based on application needs, such as individuals, communities, or organizations. Agents of the same type share behaviors and communication patterns, facilitating scalable and modular simulations.

The temporal dimension is seamlessly integrated into the overall framework through the systematic application of intra- and inter-propagation mechanisms at discrete time intervals, since the time is divided into uniform intervals in MASFENON, meaning that contact times needs to be quantized, as we will see later. This mechanism ensures that temporal dynamics are consistently captured and updated, aligning closely with observed phenomena in various domains.

Each edge is associated with contact-time attributes and represents a discrete opportunity for inter-agent communication. By leveraging this structure, the system enables a time-stepped simulation with alternated intra- and inter-propagation phases that reflect both local dynamics and global interdependence.

Virtual outputs in the source agent are linked to virtual inputs in the target agent, enabling inter-agent communication via temporal edges.

The iterative propagation algorithm that was described uses the following time parameters:

- $currentN_{inter}$: the number of the current inter-iteration;
- $currentN_{intra}$: the number of the current intra-iteration;
- N_{intra} : the maximum number of intra-iterations for each inter-iteration;
- N_{inter} : the maximum number of inter-iterations;
- $timestep$: the time between two inter-iterations.

The iteration $n \in \mathbb{N}$ is defined as: $n = currentN_{inter} * N_{intra} + currentN_{intra}$

In MASFENON, time is divided into uniform intervals, each spanning the same length ($timestep$). Therefore, the current time (t_n) at iteration n is defined as:

$$t_n = \frac{timestep}{N_{intra}} \times n$$

3.1.4 UML Diagram of the Classes

To visualize and effectively document the structural design and interactions within MASFENON, a detailed Unified Modeling Language (UML) diagram is included. This UML diagram clearly delineates the class structure, highlighting critical components such as agents, virtual nodes, propagation mechanisms, and their interrelations. The UML serves as an essential reference, facilitating understanding of the framework's internal architecture and aiding developers in extending or customizing the system for specific applications. The UML diagram is represented in figure 3.4.

The conceptual UML diagram presented here provides a high-level overview of the MASFENON framework, emphasizing the primary structural components and their interactions without focusing on detailed implementation specifics. The structure is organized into four conceptual layers, each addressing distinct aspects of the framework:

Multi-Agent System Layer This layer represents the global simulation environment.

- **MultiAgentSystem:** Manages the overall simulation process, including agent initialization, intra- and inter-propagation phases, and temporal interactions. This component orchestrates the simulation steps and ensures proper time-stepping and communication among agents.

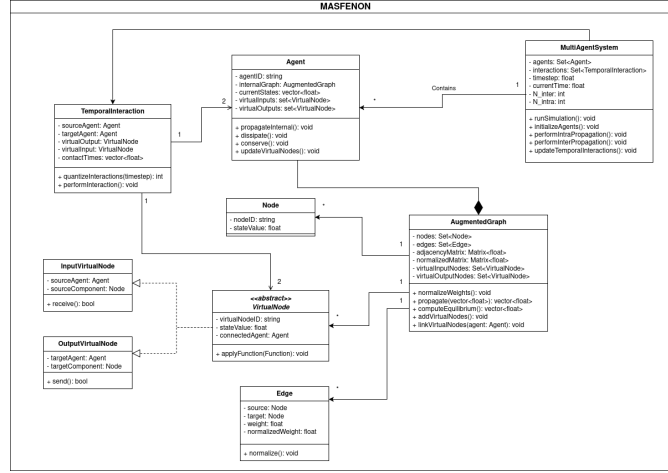


Figure 3.4: UML diagram showing the conceptual model of MASFENON

Agent Layer Each *Agent* models an autonomous entity within the multi-agent system, encapsulating its internal states, topology, and dynamics.

- **Agent:** Represents entities ranging from biological cells to organizations. Each agent maintains its internal network structure (*AugmentedGraph*), manages state dynamics (propagation, dissipation, conservation), and interacts with other agents through virtual nodes.

Network Layer This layer defines the structural and functional components within an agent's internal network.

- **AugmentedGraph:** A directed graph representing the internal structure of an agent, including both standard and virtual nodes. It supports propagation dynamics and maintains normalized adjacency matrices.
- **Node:** Represents standard nodes within an agent's internal network.
- **VirtualNode:** Abstract class, representing specialized nodes that mediate communication between agents. They explicitly link different agents and differentiate between inputs and outputs.
- **outputVirtualNode:** Implementation of the abstract class representing the virtual nodes that contain outgoing information (has a send method to send the outgoing information to target).
- **inputVirtualNode:** Implementation of the abstract class representing the virtual nodes that contain ingoing information (has a receive method to receive the information coming from another source).
- **Edge:** Defines directed, weighted connections between nodes, essential for modeling propagation dynamics accurately.

Temporal Layer Addresses the temporal dimension and discrete interactions between agents.

- **TemporalInteraction:** Represents discrete communication events, managing temporal interactions through virtual nodes. It quantizes contact events within defined intervals, ensuring accurate simulation of time-based communication.

Relationships Among Components The UML structure illustrates clear relationships:

- **Composition:** *MultiAgentSystem* composes multiple *Agents*, each of which encapsulates an *AugmentedGraph*.
- **Composition:** *AugmentedGraph* comprises *Nodes*, *VirtualNodes*, and *Edges*, defining the agent’s internal structure.
- **Association:** *TemporalInteraction* connects agents through their *VirtualNodes*, enabling inter-agent communication over time.

This UML diagram succinctly captures the modularity and conceptual clarity of the MASFENON framework, highlighting its suitability for modeling complex, dynamic, and interconnected systems.

In subsequent sections, we further elaborate on detailed implementation, computational complexity, and scalability considerations of the framework.

3.1.5 Dynamical Principles and Core Modeling Functions

The algorithmic design of MASFENON draws inspiration from deterministic chaos and complex systems theory. Concepts such as sensitivity to initial conditions, nonlinear feedback, and topological mixing are embedded in the propagation logic, enabling the simulation of both stable and oscillatory behaviors, including bifurcations and unpredictable transitions, or finding convergence behavior depending on the topology of the networks, the interactions of the agents and the propagation dynamics. These dynamics are central to modeling real-world complex systems such as biological signaling, population shifts, or cascading failures.

At the heart of MASFENON lie three transformation dynamics: *propagation*, *dissipation*, and *conservation*, which govern the intra-agent dynamics of the system. These operate on a perturbation state vector $\vec{v}^{(n)} \in \mathbb{R}^{\dim(Ag)}$ representing node-level quantities in the augmented network $\mathcal{G}_{Ag}$ associated with an agent.

Propagation The propagation mechanism in MASFENON models how influence or signals are transmitted throughout the network based on its underlying topology and the time. At each timestep, node states represented by the vector $\vec{v}$ are updated by considering the network’s connectivity, captured through a normalized adjacency matrix. This mechanism emulates how information, perturbations, or any modeled property diffuses through

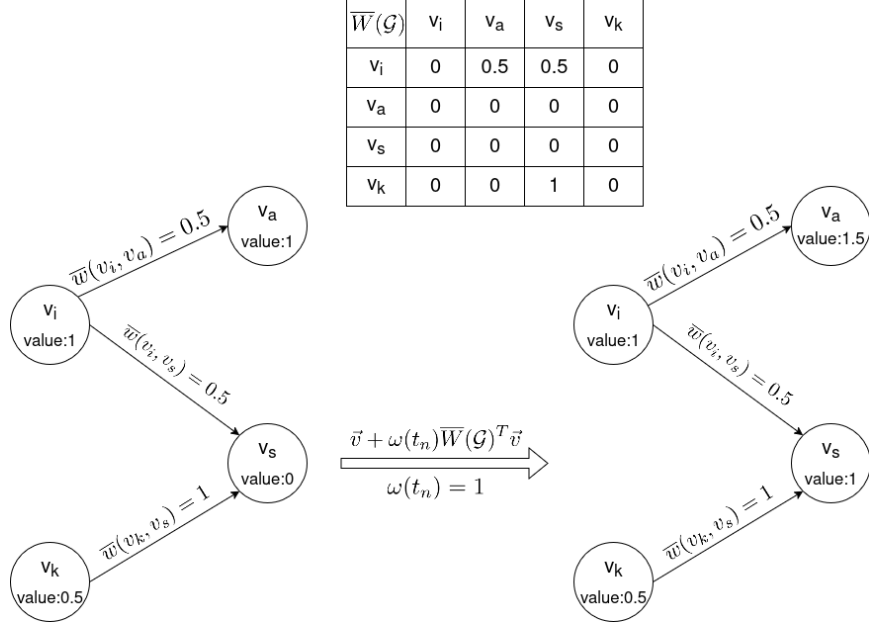


Figure 3.5: Propagation dynamics example with neighborhood based function.

interconnected nodes, such as cells in biological networks, agents in social models, or entities in communication systems. It needs to be noted that we are using a general vector rather than specifically $\vec{v}^{(n)}$. Since we are considering a general function operating on an arbitrary state vector, we do not explicitly reference the iteration step n here, while the specific state evolution at time n is considered explicitly in the complete agent dynamics.

A primary customizable propagation approach utilized in MASFENON employs direct neighborhood-based propagation. In this approach, the updated node states are computed by iteratively adding scaled influences from neighboring nodes to the current state:

$$f(\mathcal{G}, \vec{v}, t_n) = \vec{v} + \omega(t_n)\overline{W}(\mathcal{G})^T \vec{v} \quad (3.2)$$

where $\omega(t_n) \in \mathbb{R}$ is a user-defined, time-dependent scaling function controlling propagation intensity, and $\overline{W}(\mathcal{G})$ represents the row-normalized adjacency matrix of the augmented graph $\mathcal{G}$. This ensures that each node's influence is proportionally distributed among its neighbors. By customizing $\omega(t_n)$, users can adjust the speed and intensity of information diffusion within the network, allowing for the simulation of various dynamics ranging from rapid dissemination to slow, incremental propagation. An example of how the information is passed with this type of propagation can be seen in Figure 3.5

This iterative neighborhood-based propagation approach is inherently more stable, interpretable, and controllable. Stability arises because each

step adds a limited and predictable increment of influence proportional to the local network structure and scaling factor. Interpretability is enhanced, as the impact of each node’s neighborhood is explicitly and transparently accounted for at each iteration. Finally, controllability is provided by the clear and direct dependency of propagation dynamics on the scaling function $\omega(t_n)$, giving users intuitive and straightforward methods to control the overall behavior of simulations. For a complete derivation of this function from its node-level components, see Appendix A.3.

Alternatively, MASFENON provides a convergence-based propagation model, which calculates propagated states as equilibrium or steady-state solutions of the underlying network dynamics without iterative updates. Specifically, this approach computes propagation through the Moore-Penrose pseudo-inverse:

$$f(\mathcal{G}, \vec{v}, t_n) = \omega(t_n)(I - \overline{W}(\mathcal{G})^T)^+ \vec{v} \quad (3.3)$$

In this expression, $(I - \overline{W}(\mathcal{G})^T)^+$ denotes the pseudo-inverse of $(I - \overline{W}(\mathcal{G})^T)$, providing a closed-form solution that represents the equilibrium node states resulting from the initial perturbation vector $\vec{v}$. The scaling function $\omega(t_n)$ retains its role as before, modulating the intensity of the steady-state values obtained. This method can be particularly advantageous when the interest is primarily in understanding equilibrium conditions or steady-state distributions, rather than tracking transient behaviors.

However, a critical caveat of the pseudo-inverse method lies in its numerical instability. Computing the Moore-Penrose pseudo-inverse often involves singular value decomposition (SVD), which can be numerically sensitive, especially in the presence of near-singular or ill-conditioned matrices, frequently encountered in sparse or large-scale network contexts. Such instability might introduce significant numerical inaccuracies, rendering results less reliable or interpretable, especially for large systems. Therefore, users should carefully evaluate the suitability of the pseudo-inverse-based approach, particularly in high-dimensional networks where the numerical conditioning of the adjacency matrix can pose significant challenges.

In summary, the choice between these propagation methods depends heavily on the specific simulation objectives and system properties. For scenarios emphasizing controlled, interpretable, and numerically stable incremental influence updates, the iterative neighborhood-based propagation function (Eq. 3.2) is highly recommended. Conversely, when equilibrium analysis and steady-state outcomes are of primary concern, and numerical stability conditions are acceptable or can be managed, the pseudo-inverse based propagation method (Eq. 3.3) provides a powerful, though more sensitive alternative. The suggestion is to just experiment with boths and see the results, especially when simulating a real system with real data, or just to show the differences in behaviour.

It should also be noted that users have the flexibility to define their own

custom propagation functions. In particular, users can provide any complete propagation function of the form:

$$f : \mathcal{G} \times \mathbb{R}^{|V(\mathcal{G})|} \times \mathbb{R} \rightarrow \mathbb{R}^{|V(\mathcal{G})|},$$

explicitly mapping the graph topology, current node states, and simulation time to the updated node states. This flexibility represents a significant customization feature within MASFENON, allowing users to fully adapt the propagation mechanism to match specific theoretical assumptions, modeling needs, or empirical conditions relevant to their domain or application.

For further mathematical derivations and detailed explanations of these functions, please refer to Appendix A.3.

Dissipation The dissipation mechanism in MASFENON models the natural attenuation or fading of influence over time, an essential aspect for simulating realistic dynamics where signals, energy, or awareness decay progressively. This functionality aligns with deterministic chaos theory principles, particularly those involving convergence toward attractors and system stability through local contraction and decay. Dissipation is thus crucial in preventing unbounded growth in state vectors and ensuring numerical stability in iterative simulations.

A basic and commonly employed dissipation function within the framework is the exponential decay model:

$$d(\vec{v}, t_n) = \vec{v} - \lambda(t_n)\vec{v}, \quad (3.4)$$

where $\lambda(t_n) \in \mathbb{R}$ is a time-dependent parameter controlling the instantaneous rate of decay. This time-dependent parameter is also a customizable function that could be user-defined via parameters. This formulation directly scales down the state vector proportionally, modeling scenarios such as biochemical substance degradation or gradual loss of signal strength.

To enrich the dynamical complexity and explore phenomena closely associated with chaotic systems, such as sensitivity to initial conditions and periodic attractors, MASFENON supports alternative dissipation functions:

- **Periodic Dissipation:** Introduces rhythmic variations to the decay rate, allowing simulations to reflect scenarios such as seasonal or cyclic attenuation. The periodic dissipation function can be formulated as:

$$\lambda(t_n) = \lambda_0 + \lambda_1 \sin(\omega_d t_n), \quad (3.5)$$

where λ_0 is the baseline dissipation rate, λ_1 controls the amplitude of periodic modulation, and ω_d determines the frequency of the periodic decay.

- **Stochastic Dissipation:** Incorporates randomness into the dissipation process, making the system’s evolution non-deterministic and sensitive to random perturbations, enhancing realism in contexts involving noisy environments or inherent variability:

$$\lambda(t_n) = \lambda_0 + \xi_n, \quad \xi_n \sim \mathcal{N}(0, \sigma^2), \quad (3.6)$$

with ξ_n drawn from a Gaussian distribution with mean 0 and variance σ^2 , and λ_0 acting as the mean decay rate. The introduction of random noise also serves the purpose of reducing deterministic bias and can lead to more robust and unbiased simulation outcomes over longer time horizons.

These alternative dissipation mechanisms allow the simulation of a diverse array of behaviors typical in chaotic systems, including intermittent dynamics, bifurcation phenomena, and sensitivity to slight changes in initial conditions or parameters. By varying dissipation strategies, researchers can model complex interactions observed in real-world systems, ranging from biological networks with varying metabolic rates to social systems with fluctuating information decay.

The flexibility provided by these dissipation functions ensures that MASFENON remains robust and adaptable across numerous scientific and practical domains, effectively capturing the nonlinear, dynamic nature of complex interconnected networks. Like with propagation, users have the flexibility to define their own custom dissipation function as well.

For further mathematical details and theoretical explorations, particularly concerning the chaos-theoretic interpretations and stability implications, readers are encouraged to consult Appendix A.3.

Conservation The conservation mechanism in MASFENON ensures the balanced redistribution of quantities, such as information, energy, or influence, within a network after propagation events. It reflects the principle that, within a closed or semi-closed system, quantities are not arbitrarily lost or created but are instead transferred between nodes. This mechanism can therefore be interpreted as the *dual complement* of propagation: while propagation determines how information spreads across connected nodes, conservation ensures that the source node’s contribution is correspondingly reduced, maintaining a global balance. Such conservation aligns with fundamental principles found in thermodynamics and information theory, reinforcing the realistic behavior of simulations where information or resources are neither spontaneously generated nor destroyed.

In contrast, the **dissipation** mechanism operates locally at the level of individual nodes and represents a gradual intrinsic decay (or amplification, if negative) of node values over time, independent of network topology. Dissipation models irreversible processes such as degradation, noise, or external

energy loss, whereas conservation governs the redistribution of quantities during propagation.

Thus, the two mechanisms are not contradictory but complementary:

- **Dissipation** modifies node states as a time-dependent decay or gain process, independent of network interactions.
- **Conservation** enforces a redistribution constraint that preserves the total propagated quantity within the network at each iteration, counterbalancing propagation without negating dissipation.

In practice, dissipation and conservation act on different temporal and structural levels; dissipation regulates the intrinsic node dynamics, while conservation governs the inter-node balance, allowing MASFENON to flexibly represent both open and closed-system behaviors depending on model configuration.

The conservation function captures these behaviors through a weighted feedback mechanism, which modulates how much of a node’s current state is retained and how it is redistributed to connected nodes:

$$c(\mathcal{G}, \vec{v}, t_n) = \theta(t_n) \left((\overline{W}(\mathcal{G}) \vec{q}(\mathcal{G})) \odot \vec{v} \right), \quad (3.7)$$

where $\vec{q}(\mathcal{G}) \in \mathbb{R}^{|\mathcal{G}|}$ represents node-specific weighting factors that indicate how strongly each node participates in conserving and distributing its influence across the network. The scalar $\theta(t_n)$ serves as a dynamic scaling function, adjusting the intensity of conservation at each timestep, and the operation $\odot$ denotes element-wise multiplication, ensuring node-wise conservation and redistribution of influence. A thing to note is that the conservation function is not being subtracted to the input vector (as it should be since its purpose is to maintain the same amount of information in the network), that is because that function is returning the conservation term, that represents the amount of information that was passed through propagation. This term will be subtracted in the final update rule for the state of the network in Equation 3.8. An example of the effect of conservation applied to the network seen also in the example for propagation in figure 3.5 can be seen in Figure 3.6. A combination of propagation and conservation can also be seen in Figure 3.7, where the propagation spreads the information in the network, while conservation maintains the same amount of information in the whole network.

From a conceptual standpoint, the conservation dynamics can be viewed as analogous to energy conservation in physical systems, where the total energy within a closed network remains constant, merely changing form or location. Similarly, in the context of information theory, this conservation ensures that information is neither lost nor artificially amplified during interactions, but is instead systematically transferred across network connections.

By carefully tuning $\vec{q}(\mathcal{G})$ and $\theta(t_n)$, users can model scenarios where certain nodes act as influential hubs or reservoirs of conserved resources,

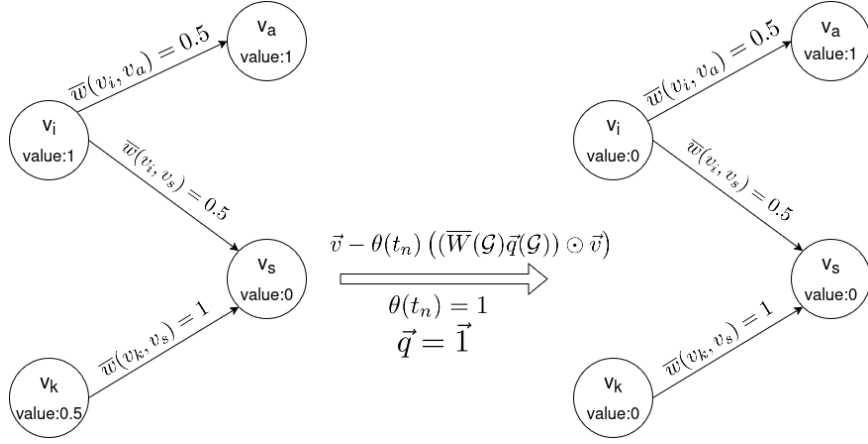


Figure 3.6: Conservation dynamics example

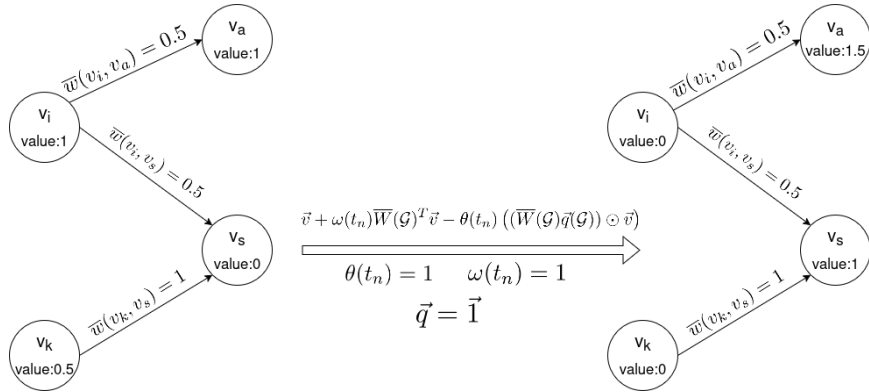


Figure 3.7: Combination of propagation and conservation dynamics in a example network

while others primarily redistribute incoming signals. This flexibility supports simulations in various domains, such as modeling metabolic networks where biochemical mass conservation is crucial, information networks where the flow of knowledge must remain coherent, or ecological systems where energy transfer and resource allocation play significant roles.

Ultimately, the conservation function enhances the realism and robustness of the MASFENON framework, enabling it to capture essential features of complex systems, preservation of core resources, coherent information dynamics, and balanced influence distribution, which are fundamental for accurately modeling real-world interconnected networks.

A further consideration regarding the propagation, dissipation, and conservation dynamics is that the scaling functions defined so far return scalar values, which are then multiplied with the corresponding dynamic term. However, the formulation also supports *node-specific* scaling parameters, allowing the scaling function to return a vector whose dimension matches the number of nodes in the target network. In this case, each node can be assigned its own scaling coefficient, enabling finer control over the dynamics.

This vectorized formulation greatly increases the adaptability of the simulation, as it allows differential tuning of parameters within the same network. Such an approach will be particularly relevant in Chapter 4, and specifically in Section 4.3, where vectorized scaling functions are exploited to fit simulation outputs to experimental data with greater granularity, improving both accuracy and flexibility of the model.

Additional details on the mathematical properties and implications of the conservation mechanism are provided in Appendix A.3.

Iterative Update Rule The iterative update rule integrates the previously described propagation, dissipation, and conservation mechanisms into a coherent dynamic system that governs the evolution of each agent’s internal state over discrete time steps. The resulting iterative process captures intricate behaviors commonly observed in complex adaptive systems, such as emergent phenomena, feedback-driven stability or instability, and energy conservation dynamics analogous to those in physical or biological networks.

Formally, the update rule can be described as follows:

$$\vec{v}^{(n)}(\mathcal{G}) = \begin{cases} f(\mathcal{G}, \vec{v}^{(0)}(\mathcal{G}), t_n) & \text{if } n = 1 \\ f(\mathcal{G}, d(\vec{v}^{(n-1)}(\mathcal{G}), t_n), t_n) - c(\mathcal{G}, d(\vec{v}^{(n-1)}(\mathcal{G}), t_n), t_n) & \text{if } n > 1 \end{cases} \quad (3.8)$$

At the first iteration ($n = 1$), the propagation mechanism initializes the state evolution by directly applying influence dynamics based on the initial conditions, representing an initial perturbation or system input, and propagating the values in the network. In subsequent iterations ($n > 1$), the frame-

work sequentially applies the dissipation function first and foremost, which models the gradual fading or controlled modulation of influence, followed by propagation to distribute the resulting state across connected nodes, and finally the conservation function, which redistributes retained influence, ensuring that the overall "information" or "energy" is conserved within the system.

The combination of these three distinct mechanisms enables MASFENON to emulate realistic and sophisticated behaviors characteristic of real-world complex systems, such as stable attractors, oscillatory dynamics, or critical transitions. The dissipation function contributes to system stability, pushing the network towards steady-state or equilibrium conditions, while the propagation function allows exploration of various transient phenomena, including spreading processes and cascading events. The conservation function complements these by guaranteeing that certain quantities remain consistently accounted for, mirroring energy or information preservation principles.

Furthermore, this iterative model inherently reflects principles of chaos theory, particularly sensitivity to initial conditions and the potential emergence of bifurcations and nonlinear dynamics, depending on the chosen parameters and functions. By finely tuning the scaling factors and parameters involved in each function, users can investigate a wide range of dynamical scenarios, from stable and predictable evolutions to highly sensitive, unpredictable, and chaotic behavior.

Additionally, given that interactions between agents in the inter-propagation phases occur at discrete, defined temporal intervals (contact times), the framework implements a quantization strategy for accurately representing these interactions within simulation intervals. Multiple discrete contacts occurring within a single time interval can be modeled either as a single consolidated interaction or as a cumulative interaction weighted by the number of discrete events, as illustrated in Fig. 3.8.

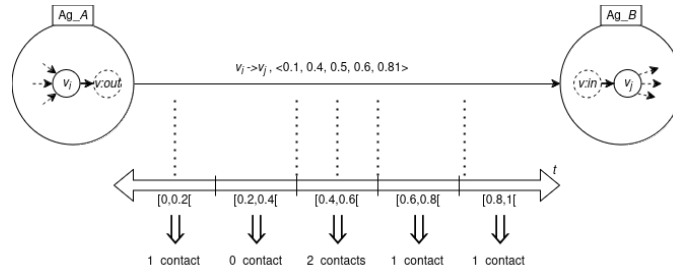


Figure 3.8: Contact quantization for an edge in the *Agents* temporal network. In this example, the chosen timestep is 0.2, and inter-agent communication is quantized accordingly.

The two quantization approaches provided by MASFENON are:

$$\#contacts(t_i, t_{i+1}) = \begin{cases} 0 & \text{edge is not considered,} \\ 1 & \text{edge is considered.} \end{cases} \quad (3.9)$$

and

$$\#contacts(t_i, t_{i+1}) = \begin{cases} 0 & \text{edge is not considered ,} \\ n, n \geq 1 & \text{edge is considered, weight * } n \end{cases} \quad (3.10)$$

These quantization strategies further enhance the flexibility and realism of the MASFENON framework by allowing precise modeling of temporal dynamics at varying resolutions, capturing nuances of interaction frequencies and intensities in complex interconnected systems.

The Eq. 3.10 considers the multiple contact times in the interval as the force of interaction of the inter-propagation. For more information about the mathematical background of this contact quantization, see the Appendix A.3

The choice of temporal granularity and quantization strategy has direct implications on the resolution, fidelity, and computational behavior of the simulation. Specifically, the number of intra-agent iterations N_{intra} determines the precision and speed of state evolution within each agent. Increasing N_{intra} allows finer-grained internal dynamics, enabling rapid convergence or adaptation to perturbations over a single macro-time interval. In contrast, the number of inter-agent iterations N_{inter} , in combination with the chosen *timestep*, controls the overall simulation length and the spacing between inter-agent communication events.

Moreover, smaller timesteps combined with larger N_{intra} values yield higher resolution in the quantization process, enabling the system to resolve subtle changes in contact frequency. This configuration allows individual contact events to be isolated within a single interval or be sparsely distributed across time, resulting in some intervals exhibiting no inter-agent communication at all. As a result, temporal sparsity and burstiness can be accurately represented, reflecting real-world phenomena such as asynchronous signaling or episodic behavior.

Therefore, careful tuning of these temporal parameters is crucial: they directly influence the model's ability to simulate fast versus slow processes, control computational complexity, and determine whether certain dynamical effects are observable. Proper selection of N_{intra} , N_{inter} , and *timestep* is essential for ensuring both the numerical stability and the biological or domain-specific plausibility of the simulation outcomes. We will see shortly these parameters and their role in the temporal characterization of the framework.

Inter-Agent Communication While intra-agent dynamics are governed by the detailed propagation, dissipation, and conservation mechanisms outlined previously, interactions *between* agents are explicitly modeled through an inter-agent communication mechanism employing virtual nodes. At specific intervals determined by the inter-propagation scheduling parameters, the states of virtual output nodes from source agents are systematically transferred to the corresponding virtual input nodes of target agents. Formally, the state update for virtual nodes at iteration n is expressed as follows:

$$\vec{v}_x^{(n)}(\mathcal{G}_{\text{target}}) = \vec{v}_y^{(n)}(\mathcal{G}_{\text{source}}) \quad (3.11)$$

Conceptually, this approach can be viewed as a structured information-passing mechanism, where virtual nodes act as *gateways* facilitating regulated interactions and ensuring that information exchange is both directional and temporally discrete. The use of virtual nodes ensures that each agent maintains clear boundaries of internal dynamics, while still enabling sophisticated external communication and feedback from other agents.

Moreover, this design supports layered, time-resolved modeling of complex interdependencies, accommodating both direct interactions and indirect or latent influences. It also provides significant flexibility, allowing users to precisely define and control communication patterns, such as periodic synchronization, intermittent interactions, or event-driven information exchanges. Thus, the inter-agent communication mechanism is crucial for accurately capturing multi-scale dynamics and emergent behaviors within the interconnected system modeled by MASFENON.

Temporal Execution Loop The simulation workflow in MASFENON proceeds through a structured temporal execution loop, integrating intra-agent and inter-agent propagation phases. Specifically, each simulation iteration is composed of a nested two-level loop: an inner loop handling intra-agent propagation, and an outer loop managing inter-agent propagation. After precisely N_{intra} iterations of intra-agent state updates, representing the detailed internal dynamics and local information diffusion within each agent, the inter-agent propagation phase is activated, facilitating the global exchange of information and influence among agents through virtual node connections. This iterative nested structure ensures a seamless interplay between the agents’ internal evolution and their interactions with the broader agent network.

Edges connecting *Agents* within the temporal multi-graph are explicitly associated with real-valued contact times, representing discrete opportunities for interaction. The propagation dynamics across these edges rely on virtual nodes, ensuring information or influence transfer from the virtual outputs of source agents to the corresponding virtual inputs of target agents. Such temporal edges enable the simulation of realistic temporal interaction patterns, including periodic, sporadic, or event-driven exchanges.

The core temporal parameters governing the execution loop are rigorously defined as follows:

- $N_{\text{intra}} \in \mathbb{N}$: Defines the granularity of internal state updates for each agent between global communication events (inter-iterations).
- $N_{\text{inter}} \in \mathbb{N}$: Establishes the number of global communication events, effectively controlling the overall length of the simulation.
- $\text{current}N_{\text{inter}} \in \{0, 1, \dots, N_{\text{inter}}\} \subset \mathbb{N}$: Tracks the current iteration within the global inter-agent propagation loop.
- $\text{current}N_{\text{intra}} \in \{0, 1, \dots, N_{\text{intra}}\} \subset \mathbb{N}$: Tracks the current iteration within the local intra-agent propagation loop.
- $\text{timestep} \in \mathbb{R}^+$: Determines the length of time intervals between successive global communication events, shaping the temporal scale of the simulation.

Each iteration $n \in \mathbb{N}$ of the overall simulation is precisely computed by combining the counts of intra- and inter-propagation loops:

$$n = \text{current}N_{\text{inter}} \cdot N_{\text{intra}} + \text{current}N_{\text{intra}} \quad (3.12)$$

Correspondingly, the simulation's temporal dimension is partitioned uniformly into discrete intervals, each of duration $\frac{\text{timestep}}{N_{\text{intra}}}$, allowing high temporal resolution and flexibility:

$$t_n = \frac{\text{timestep}}{N_{\text{intra}}} \times n \quad (3.13)$$

The careful selection of these parameters shapes not only the computational performance of the simulation but also its capacity to capture complex dynamical behaviors and emergent patterns. By clearly defining the relationship between local dynamics and global communication, the temporal execution loop ensures numerical stability, interpretability of results, and realistic representation of processes across multiple scales and domains.

3.2 Implementation Details

The implementation of the MASFENON framework is an integral component of both its theoretical foundations and its practical applicability. A detailed examination of its implementation is crucial, as many of the theoretical concepts, particularly those related to scalability, propagation dynamics, and inter-agent interactions, are inherently intertwined with choices made during software design and engineering. Thus, discussing implementation details is not merely an exercise in technical documentation but rather a critical exploration of how computational and engineering constraints have shaped the theoretical development and methodological choices behind MASFENON.

Given the computational intensity involved in simulating complex, interconnected networks across multiple temporal and spatial scales, the design

of MASFENON explicitly leverages advanced high-performance computing (HPC) techniques. Implementing the framework in C++ and utilizing the Message Passing Interface (MPI) standard for distributed computing allowed not only efficient handling of large datasets and heavy computational loads but also shaped the theoretical modeling approaches, such as the choice of propagation mechanisms, quantization strategies, and state-space representation, to be inherently compatible with distributed and parallel processing paradigms.

Moreover, rigorous attention has been given to software engineering best practices essential for large-scale and distributed systems, including modularity, extensibility, and careful management of memory and computational resources. Matrix and linear algebra operations, which are central to propagation computations, have been parallelized using robust HPC libraries such as Armadillo and OpenMP.

Distributed and reactive programming paradigms have been employed to manage agent communications and interactions effectively, reflecting a theoretical emphasis on adaptability and responsiveness in multi-agent simulations.

In the following subsections, we detail how various implementation aspects, such as framework customization capabilities, inter-agent communication via OpenMPI, intra-agent computation with Armadillo and OpenMP, scalability considerations, optimization techniques, and rigorous memory management strategies, collectively enable MASFENON to efficiently simulate and analyze the complex dynamics of large-scale, temporally evolving multi-agent networks.

3.2.1 Customization of the framework

A defining feature of the MASFENON framework is its high degree of configurability, allowing users to tailor the simulation process to their specific modeling requirements. This customization spans data input structure, modeling logic, mathematical dynamics, output formatting, and performance control, making the framework suitable for both exploratory and large-scale, production-level simulations. These options are particularly important in high-performance computing environments, where flexibility must coexist with efficiency and reproducibility. Before going through the single parameters, it should be noted that the most important parameters used to tune the model are the scaling functions and their parameters, along the possibility of having some custom user-defined dynamics.

From a modeling perspective, several parameters enable detailed control over the simulation logic. These include the definition of input perturbation vectors (**initialPerturbationPerType**, **initialPerturbationPerTypeFolder**), subtyping and node metadata (**subtypes**, **nodeDescriptionFile**, **nodeDescriptionFolder**), and the graphs that define agent topolo-

gies (**graphsFilesFolder**). Inter-agent connections and structural heterogeneity are encoded using interaction folders and virtual node configurations (**typeInteractionFolder**, **virtualNodesGranularity**).

Temporal dynamics are controlled through the definition of the number of intra- and inter-type iterations (**intratypeIterations**, **intertypeIterations**) and their associated time discretization (**timestep**). These are central to the temporal loop and quantization schemes discussed in Section 3.1.5 and are key to balancing precision and simulation runtime.

In addition to structural input, MASFENON allows full customization of the mathematical modeling functions. Users can specify the dissipation model (**dissipationModel**, **dissipationModelParameters**, **dissipationModelParameterFolder**), conservation model (**conservationModel**, **conservationModelParameters**, **conservationModelParameterFolder**), and propagation model (**propagationModel**, **propagationModelParameters**, **propagationModelParameterFolder**) from a range of built-in options. These options reflect theoretical considerations like energy decay, stochasticity, conservation principles, and equilibrium computation, offering control over both transient and steady-state dynamics. As it was said before, these are the most important parameters to tune the framework and making it customizable for different use cases, working in tandem with real data.

These built-in options for propagation, dissipation and conservation are the following:

- **Propagation models:** `default`, `neighbors`, `scaled`, `customScaling`, `customScalingNeighbors`, `customPropagation`
- **Dissipation models:** `none`, `power`, `periodic`, `random`, `scaled`, `custom`
- **Conservation models:** `none`, `scaled`, `random`, `custom`

In addition to selecting among pre-implemented models, users can directly modify the codebase to implement their own custom modeling functions. This is particularly useful in cases where domain-specific behavior or novel dynamics are desired.

Each modeling function also supports a user-defined time-dependent scaling function:

- $\omega(t_n)$: controls the intensity of propagation.
- $\lambda(t_n)$: controls dissipation (decay) rate.
- $\theta(t_n)$: modulates conservation intensity.

These scaling functions can be customized by the user through parameter inputs or by defining new functions in the codebase. This flexibility allows for a wide range of temporal behaviors, including deterministic decay, oscillatory dynamics, and stochastic fluctuations.

At the current stage, the customization of modeling functions is performed by editing and recompiling the C++ source code. Although this provides full control over the simulation logic, it requires familiarity with the codebase and build system.

Future development plans include introducing a plugin-based architecture based on runtime dynamic linking. This would allow users to define and compile custom modules as shared libraries, which can be loaded at runtime without recompiling the entire framework. Such a feature would enable rapid prototyping, greater extensibility, and improved modularity, facilitating easier experimentation and broader community contributions.

The current implementation of MASFENON provides:

- Extensive control over the propagation, dissipation, and conservation dynamics.
- Customization of associated scaling functions.
- Full access to implement new logic through source code modifications.

While this enables deep theoretical exploration and simulation fidelity, upcoming features will improve runtime flexibility and usability for a broader user base. We will see some options of implementation for additional customization in chapter 5.

Several additional flags support further specialization. For example:

- **sameTypeCommunication** enables communication between agents of the same type by introducing internal virtual nodes (self-loops in the agent network).
- **saturation**, **saturationTerm**, and **customSaturationFunction** define constraints on perturbation values, ensuring bounded simulations.
- **conservateInitialNorm** enforces the conservation of the initial Euclidean norm of node states throughout iterations.
- **quantizationMethod** governs how temporal edge contacts are interpreted when multiple contacts fall within the same timestep, as seen before in section 3.1.5

Saturation is a very important parameter that can be personalized for each node in every single agent. It practically forces a convergence of the state values of the network, and is very useful in case of bounded simulations. For some examples, see Chapter 4 where we use the saturation in the context of epidemics. Also, see the Appendix for some additional information on how saturation is performed.

From an HPC perspective, logging and execution control are equally important. Options like **verbose**, **loggingOptions**, **treatWarningAsError**, and **resumeCheckpoint** allow for flexible debugging, monitoring, and fault tolerance in distributed simulations. Output and performance tracking can be tuned through parameters such as **outputFolder**, **outputFormat**, **savePerformance**, and **saveAugmentedNetworks**.

By exposing all these options via command-line configuration and internal configuration files, MASFENON supports a principled and modular approach to simulation control. This ensures that researchers and developers can both reproduce specific experiments and explore a wide spectrum of possible model behaviors with minimal friction, all while maintaining compatibility with the HPC runtime environment.

3.2.2 Use of OpenMPI for Inter-Agent Communication

Inter-agent communication in MASFENON is implemented using the Message Passing Interface (MPI), specifically leveraging the OpenMPI library [32]. This choice is foundational for enabling distributed simulations where each agent, or group of agents, may be assigned to different computing nodes or cores. OpenMPI provides mechanisms for point-to-point communication, collective operations, and synchronization, which are employed in MASFENON to propagate information between agents at predefined inter-propagation intervals.

Communication between agents is efficiently managed using virtual nodes, whose state is transmitted across processes using `MPI_Send` and `MPI_Irecv` primitives. These non-blocking alternatives allow the framework to carry out computations while awaiting data from other agents, avoiding communication bottlenecks and enabling asynchronous interaction coordination. This design supports fine-grained communication scheduling and preserves the agent-based semantics by encapsulating each agent’s state in process-local memory.

Each agent operates as an autonomous unit within a workgroup, and every workgroup is assigned to a specific MPI process. In this distributed model, each MPI process typically handles one or more agents, and the number of agents per process defines the computational workload assigned to each processor:

$$workload = \left\lceil \frac{|V(\text{Agents})|}{n_{\text{processors}}} \right\rceil \quad (3.14)$$

This strategy aligns each MPI process with physical or virtual nodes of a computing cluster, ensuring effective use of available resources. In the case of uneven divisions, the last processor may receive fewer agents than others. This slight imbalance does not significantly degrade performance but opens opportunities for further optimizations such as dynamic workload balancing or intra-process communication refinement.

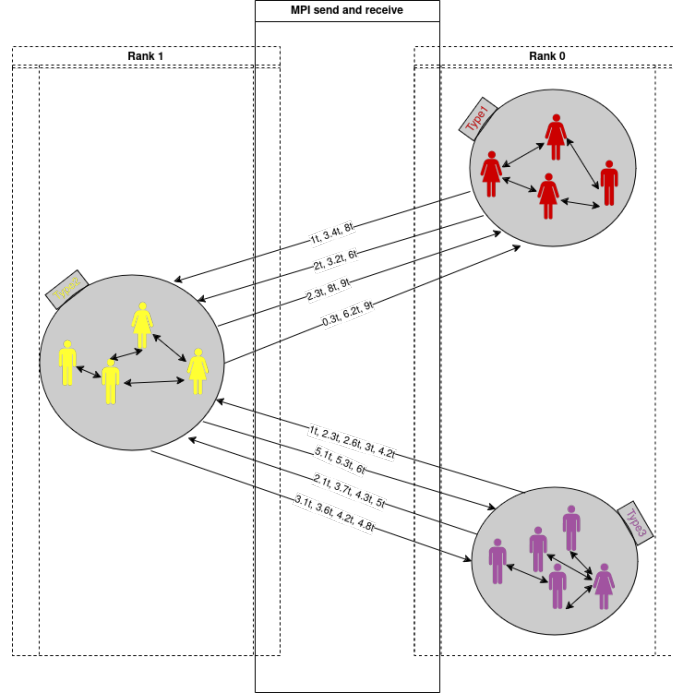


Figure 3.9: MASFENON MPI-based MAS implementation. Agents are grouped into workgroups assigned to MPI processes (ranks). In this use case, we see the nodes as single people, and each type as a group/community; they communicate through virtual nodes.

The framework minimizes inter-process communication costs through several strategies:

- Compressing exchanged state vectors where applicable;
- Overlapping computation with communication using non-blocking MPI routines;
- Batching communication only at inter-propagation steps to reduce frequency.

This hybrid parallel architecture, using MPI for inter-agent and inter-process communication and OpenMP (as discussed later) for intra-agent parallelism, allows MASFENON to scale across multiple cores and nodes. It enables simulations of large, temporally dynamic networks with performance that remains robust across heterogeneous system configurations.

3.2.3 Armadillo and OpenMP for Intra-Propagation

Intra-agent computations, specifically propagation, dissipation, and conservation, are computationally intensive due to repeated matrix-vector operations over the internal state of each agent's network. MASFENON utilizes the Armadillo C++ linear algebra library[80], which provides a clean,

MATLAB-like syntax and leverages BLAS and LAPACK backends for optimized numerical routines.

To further accelerate computations, OpenMP directives are used to parallelize loops and matrix operations within each agent. This hybrid strategy ensures:

- Efficient CPU utilization on multi-core architectures.
- Significant speedup in simulation steps, especially in large networks with dense topologies.
- Minimal code complexity due to Armadillo’s high-level abstraction and OpenMP’s incremental integration.

Armadillo’s integration is particularly valuable for expressing the core update formulas (e.g., Eq. 3.2, 3.7) in compact, readable code, while OpenMP boosts runtime performance without requiring manual thread management.

3.2.4 Scalability Considerations and Optimization Techniques

Achieving high scalability and performance in MASFENON has driven several critical design decisions and low-level optimizations:

- **Hybrid parallelism:** MPI handles inter-agent distribution, while OpenMP parallelizes intra-agent computation. This dual-layered approach maximizes hardware utilization across clusters.
- **Memory locality:** Agent-specific data is stored contiguously in memory to improve cache efficiency, particularly for repeated matrix-vector operations.
- **Load balancing:** Agents are evenly distributed across MPI ranks based on topology size and expected computational load, preventing imbalances that would degrade scalability.
- **Communication throttling:** Inter-agent propagation steps are discretized to reduce excessive messaging frequency, thereby improving network throughput.

Scalability tests have shown that MASFENON maintains strong and weak scaling efficiency across a broad range of agent configurations, making it suitable for simulations involving thousands of agents and millions of network interactions.

To see the results of the scalability experiments, see section 4.1.

3.2.5 Programming Practices and Development Workflow

To ensure sustainability, maintainability, and reliability in MASFENON’s HPC-level codebase, rigorous software engineering and DevOps practices were adopted from the beginning of the project.

Version Control and Branching All source code and configuration files are maintained under Git-based version control (e.g. GitHub), following a

branching strategy with separate 'main', 'documentation' and feature branches. This enables organized development, issue tracking, and safe merging, critical for collaborative scientific software.

Continuous Integration (CI) & Continuous Deployment (CD) Automated CI pipelines are triggered on every commit or pull request. These pipelines perform clean builds in isolated environments, execute unit and integration tests, and conduct static analysis to detect regressions early and ensure reproducibility and code quality. These pipelines also rebuild the documentation that is available at <https://josura.github.io/MASFENON/>

Clean Code and Coding Standards The architecture of MASFENON embraces established software engineering best practices to ensure a maintainable, extensible, and high-quality codebase. Specifically, the framework adheres to principles such as:

- **KISS (Keep It Simple, Stupid):** Designs and code avoid unnecessary complexity, every abstraction or feature exists only if demonstrably required, helping reduce cognitive load and potential errors.
- **DRY (Don't Repeat Yourself):** Shared logic is centralized to prevent duplication and the maintenance burden of updating multiple code paths.
- **YAGNI (You Aren't Gonna Need It):** Features and extensions are only implemented when there is immediate value, preventing over-engineering and enabling flexible future adaptations .
- **Separation of Concerns:** Responsibilities are clearly partitioned across modules, core simulation logic, propagation models, I/O handling, and parallelization layers are kept distinct, enhancing readability and modular evolution.
- **SOLID Principles:**
 - *Single Responsibility (SRP):* Components like logger, propagation engine, and MPI handler each perform a single clearly scoped role.
 - *Open-Closed (OCP):* New propagation, dissipation, or conservation models can be added by extension rather than modifying core classes.
 - *Liskov Substitution (LSP):* Subclasses or alternative implementations consistently satisfy existing contracts, enabling interchangeable agent or propagation model implementations.
 - *Interface Segregation (ISP):* Interfaces remain narrowly scoped so that modules depend only on the abstractions they actually consume.
 - *Dependency Inversion (DIP):* High-level modules depend on abstract interfaces rather than concrete classes, e.g. the simulation engine interacts via abstraction layers over low-level MPI or Ar-

madillo operations.

Moreover, consistent naming conventions, inline documentation for complex logic, and modular decomposition reinforce maintainability. This disciplined approach ensures MASFENON remains robust, adaptable, and accessible for future development, even by contributors external to the original team.

Design Patterns To support clean, maintainable, and extensible implementation of the MASFENON framework, several well-known object-oriented design patterns were employed.

Thread-Safe Singleton is used to implement the logging facility in MASFENON, so that all parts of the application share a single logging instance. Using the Meyers-Singleton idiom (a 'static' local variable in C++11 or later) provides **thread-safe initialization** without requiring manual locking or constructs. The Logger also uses a mutex to control the access to IO and make the logging more easy to handle without any conflicts between different processors. This ensures that concurrently running threads can safely log to a central destination without creating multiple logger instances or intersecting the logs in the same instance (e.g. the console).

To integrate the Armadillo linear algebra library with MASFENON's internal data structures while hiding Armadillo-specific APIs, we employ the **adapter pattern**. The adapter classes wrap Armadillo matrices and vectors, presenting a uniform internal interface to the rest of the system. This structural pattern allows code to remain decoupled from external library dependencies and facilitates future swapping or upgrading of the linear algebra backend without major refactoring (this was done because one of the possibilities of MASFENON for the future is to be almost completely independent from external libraries).

Complex subsystems such as the propagation logic, agent initialization, configuration parsing, and simulation loops are hidden behind **facade classes** that present simplified, coherent APIs to higher-level modules. This reduces coupling between modules and isolates complexity in well-defined interfaces. Interfaces (pure abstract classes) are used extensively to define the contracts for propagation, dissipation, and conservation functions, allowing multiple implementations to be plugged in cleanly.

Along with these, the code uses dependency injection extensively, along some minor patterns like **proxies**, **templates** and **state** to handle the complex dynamics of MASFENON in a modular and easily adaptable way.

The combination of all of these supports SOLID principles (e.g. interface segregation, dependency inversion) and makes the system modular, testable, and extendable.

Automated Testing A suite of unit tests (using frameworks like Google Test) and functional tests validate individual modules (propagation, dissipation, conservation routines, MPI communication) as well as end-to-end simulation runs. These are run automatically by the CI pipeline on each build.

Dependency and Build Management CMake is used to manage builds, external dependencies, and configurations. Dependencies are version-pinned and reproducible to avoid "it works on my machine" issues. Builds are reproducible and consistent across development, CI, and HPC cluster environments.

DevOps and Deployment CI workflows are declared as code (github workflows). Builds run in clean environments, and artifacts are promoted across pipeline stages without rebuilding. Rollback strategies and artifact versioning are supported to ensure stability and traceability. Deployment of documentation and software packages is also done via workflows.

Code Review and Issue Tracking All changes go through pull-request review cycles. Issues are tracked via GitHub ticketing with branches linked to issues/sub-issues, enabling structured collaboration and historical traceability. Although the main development was almost always done by me, this kind of good practice was still maintained for better handling of problems, features, and documentation.

Documentation and Code Coverage Documentation (user guides, API references, developer notes) is versioned alongside code. Automated coverage tools ensure consistent test coverage and highlight untested pathways.

Benefits and Relevance to Scientific HPC Software These practices support reproducible and reliable scientific computing. They are especially vital for long-running, distributed simulations where subtle bugs can compromise results. They also smooth collaboration across contributors and ensure that theoretical design decisions (e.g. modular propagation models, customizable scaling functions) remain tractable and well-tested in a distributed HPC context.

Future work may integrate runtime extensibility (e.g. compile-time reflections, that will be available from C++26) to further enhance customization without recompilation.

3.2.6 Control Over Memory Leaks

Memory management is a foundational concern in any high-performance simulation framework. In MASFENON, memory reliability is ensured through

a two-tier strategy:

1. **Code discipline and static analysis:** The core C++ implementation adheres to RAII (Resource Acquisition Is Initialization) principles wherever possible. Manual memory management is reviewed through static analysis tools such as *cppcheck*¹. The codebase is also thoroughly debugged and controlled via testing, especially for memory leaks or access to memory that shouldn't happen in the code.
2. **Dynamic profiling with Valgrind:** The framework undergoes regular memory profiling using the Valgrind instrumentation tool suite², particularly *memcheck*, which detects memory leaks, buffer overflows, and invalid accesses, also profiling in both a controlled environment in single or multi-core (on my computer or in a container), and in the deployment environment (HPC supercomputer).

All dynamically allocated data structures, including agent networks and message buffers, are systematically released at the end of simulations, ensuring long-running simulations remain stable and leak-free.

Future iterations of MASFENON will explore integrating more advanced memory allocators and implementing resource-pooling strategies to further reduce memory overhead and fragmentation.

Future profiling and optimization efforts are planned to continually enhance MASFENON's performance, ensuring its suitability for increasingly complex simulations and large-scale computational environments. Additional improvements

3.3 Computational Complexity and Performance Analysis

This section outlines the computational complexity of the algorithms underlying the MASFENON framework and presents initial performance analysis based on existing experiments. Further profiling and optimization, including GPU acceleration and advanced parallelization strategies, are planned for future work.

3.3.1 Time Complexity

The computational complexity of the MASFENON framework primarily hinges on the matrix multiplication operations involved in simulating propagation dynamics across interconnected networks. The complexity is most pronounced during the intra-propagation mechanism, where subgraph adjacency matrices undergo multiplication to capture the influence of neighboring nodes.

¹<http://cppcheck.net/>

²<https://valgrind.org/>

The critical operation driving computational complexity is the multiplication of subgraph adjacency matrices. Considering a graph with n nodes, the size of the adjacency matrices is $n \times n$. Therefore, the complexity of matrix multiplication is traditionally $O(n^3)$.

As the size of the graph (n) increases, the cubic nature of n^3 complexity becomes a significant factor in determining the computational load. This complexity arises during the propagation steps, where the influence of each node on its neighbors is computed.

Efforts have been made to optimize the computational performance, particularly focusing on the matrix multiplication steps. These leverage on parallel computing library (i.e. Armadillo) together with parallel programming standards OpenMP and MPI. This allows us to exploit multicore architectures, mitigating the impact of $O(n^3)$ complexity and consequently enhancing the overall computational efficiency.

Given the computational demands, scalability becomes a key consideration. While $O(n^3)$ complexity poses challenges, MASFENON is designed to harness parallel processing capabilities, enabling the framework to scale efficiently with increasing graph sizes.

As we acknowledge the inherent computational complexity, ongoing research explores avenues for further optimization and potential algorithmic enhancements, especially considering the use of Armadillo for linear algebra (using LAPACK³ and an implementation of blas, for example, cuBLAS⁴) that limits our freedom in additional optimization on memory and the use of GPUs. Profiling of the CPUs-GPUs suggests that the hardware is used but seems underloaded during transfers between memory and when sending data with MPI. Future iterations of MASFENON may introduce refined algorithms or leverage advancements in parallel computing technologies to address scalability concerns, especially in the face of the continuous extension of the framework capabilities and adaptability.

3.3.2 Space Complexity

The space complexity of the framework is mainly based on the order of the graphs passed as the input, the number of graphs, and the interaction between the different networks (the size of the inter-communication network between the agents). Formally, given the maximum order among the graphs passed as input $\max|V|$, the number of graphs $|V(Agents)|$ and the size of the inter-communication network $|E(Agents)|$, the space complexity is:

$$O \left(\left(\max|V| + \frac{|E(Agents)|}{|V(Agents)|} \right)^2 \times |V(Agents)| \right) \quad (3.15)$$

This space complexity is composed of the following parts:

³<https://netlib.org/lapack/>

⁴<https://docs.nvidia.com/cuda/cublas/index.html>

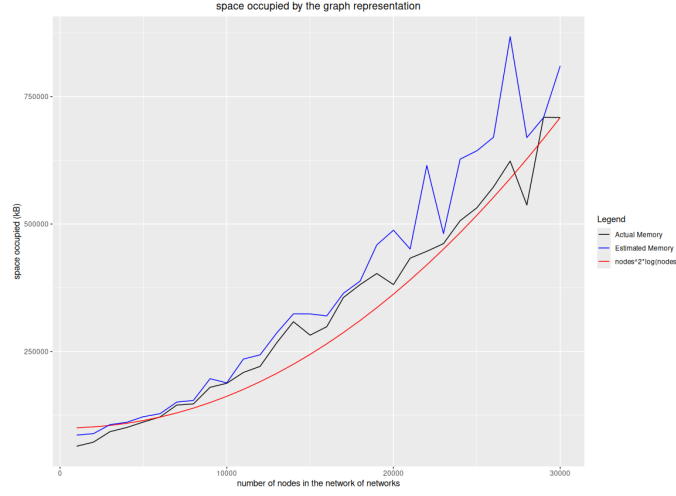


Figure 3.10: Memory used for the total number of nodes for every problem, the blue line represents the estimated memory using the formula seen in 3.15, the red line represents the function $x^2 \log(x)$

- $\left(\max|V| + \frac{|E(Agents)|}{|V(Agents)|} \right)$ is the maximum amount of nodes that can be contained in an augmented graph;
- $\left(\max|V| + \frac{|E(Agents)|}{|V(Agents)|} \right)^2$ is the amount of memory needed for an adjacency matrix;
- $O \left(\left(\max|V| + \frac{|E(Agents)|}{|V(Agents)|} \right)^2 * |V(Agents)| \right)$ is the maximum amount of memory needed for all the adjacency matrices.

This relationship indicates the memory requirements scale with the number of agents and complexity of interconnections.

As some confirmation to this space complexity formula, the memory occupied by changing the number of nodes in the setup seen for weak and strong scaling can be seen in Fig. 3.10 .

3.3.3 Parallelization Techniques and Efficiency

The computational efficiency and scalability of MASFENON rely on a hybrid parallelization strategy combining distributed and shared-memory approaches to fully exploit modern HPC architectures. The framework integrates:

- **Distributed Parallelism with MPI:** Inter-agent communication is parallelized using the Message Passing Interface (MPI), implemented via OpenMPI. Each agent, or group of agents (based on the workload size for each process, as previously seen), is assigned to a separate MPI process, allowing for distributed simulation over cluster nodes. Communication between agents (especially for inter-propagation steps) is

realized through non-blocking communication primitives like `MPI_Isend` and `MPI_Irecv`, enabling asynchronous data exchange and overlapping of communication and computation.

- **Shared-Memory Parallelism with OpenMP:** Intra-agent computations, particularly matrix operations for propagation, dissipation, and conservation, are parallelized using OpenMP. This includes:
 - `#pragma omp parallel` for loop-level parallelism during vector and matrix iterations.
 - **Parallel lambda expressions** for standard library containers (e.g., `std::vector`) to apply concurrent updates to node-level data.
 These optimizations ensure full utilization of multi-core CPUs within each node and minimize bottlenecks arising from sequential operations.
- **Linear Algebra Backend with Armadillo:** Armadillo is used as the core linear algebra library. It offers high-level syntax and efficient backends (e.g., OpenBLAS, LAPACK), and supports GPU acceleration via CUDA when available. While earlier iterations of MASFENON explicitly used CUDA for certain matrix operations, the adoption of Armadillo was motivated by its robust implementation of advanced matrix methods, particularly the Moore-Penrose pseudo-inverse, which is central to some propagation models.
- **GPU Acceleration (Under Evaluation):** While Armadillo can leverage CUDA through integration with GPU-capable backends, profiling GPU utilization in the current framework version is still ongoing. Future optimization efforts may include explicit CUDA offloading or integration with GPU-aware MPI libraries for further performance gains.

Results of the performance and speedup that we get from parallelism can be seen in Figure 4.6a and 4.6b in section 4.1.

This dual-level parallelism enables MASFENON to achieve strong scaling across nodes while maintaining efficient intra-node performance. The design ensures that both communication overhead and computation latency are minimized, resulting in a framework well-suited for simulations involving thousands of agents and large-scale dynamic graphs.

Future work will focus on detailed profiling, particularly regarding GPU utilization and optimized communication patterns, further enhancing MASFENON’s computational efficiency and scalability. This will also be explored in chapter 5 .

3.4 Additional Functionalities of MASFENON

Beyond its core simulation engine, MASFENON includes a suite of auxiliary functionalities designed to enhance the usability, interpretability, and integration of the framework with real-world data and analytical workflows.

These tools support both pre-simulation graph analysis and post-simulation interpretation and validation.

3.4.1 Graph Statistics and Structural Metrics

As part of the preprocessing pipeline, MASFENON includes the capability to compute various graph statistics and structural properties of the input networks associated with each agent. These metrics allow users to assess the topology and complexity of their input graphs prior to simulation and to validate assumptions about network connectivity. They also are really important when having different type of graph associated to the agents, since the topology of an agent heavily influence its behavior and evolution. Available metrics include, but are not limited to:

- Node and edge counts
- Degree distributions (in/out/total)
- Clustering coefficients
- Connectivity and component analysis
- Graph density and sparsity
- Centrality measures (e.g., betweenness, closeness, eigenvector)
- etc...

This module provides a comprehensive overview of the structural characteristics of each agent's internal network, facilitating comparative studies and network diagnostics. All the metrics that are available in MASFENON can be seen in the appendix A.2.

3.4.2 Visualization and Output Analysis Tools

To support the interpretation of simulation outputs, MASFENON includes a flexible set of visualization and analysis tools:

- A lightweight **web-based visualization interface**, implemented with Flask and Plotly, allows users to interactively inspect the topology of each agent's network and observe how it evolves over time. The interface provides intuitive access to simulation states at each iteration, enabling exploratory analysis and debugging.
- A set of **Python, R, and Bash scripts** enables statistical analysis and plotting of the output time-series data. These scripts can be used to:
 - Compare simulated dynamics against real or synthetic ground-truth data.
 - Generate statistical summaries and visual representations (e.g., line plots, heatmaps).
 - Automate batch comparisons across multiple simulation runs.

An example of the web visualization for some metabolic network data (that we will see in chapter 4) can be seen in Figure 3.11. Further develop-

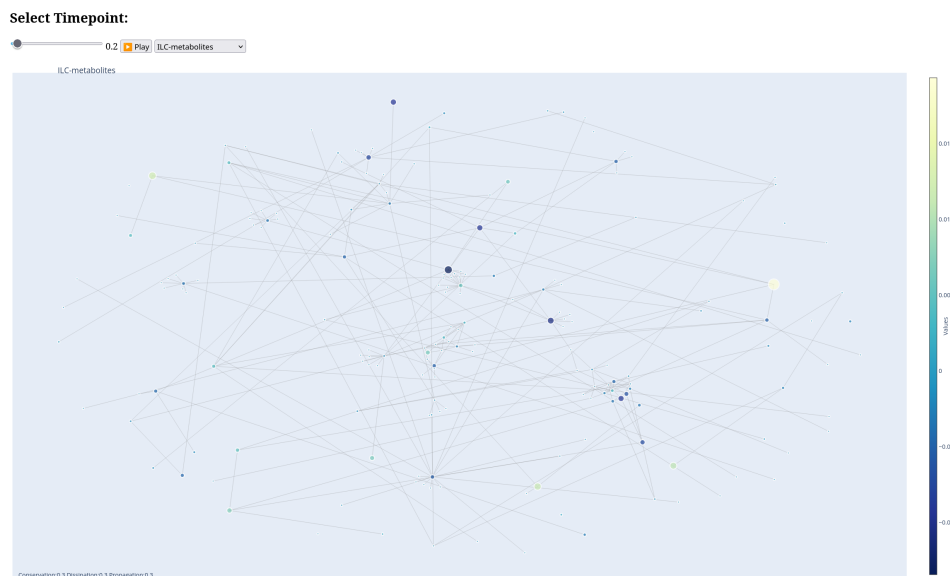


Figure 3.11: Visualization example with one of the metabolic network from the simulation that we will see in section 4.3. Different nodes can be explored, showing their names and their current state. Subgraphs can be selected as well, to show a part of the network. The size and color of the nodes is dependent on its perturbation value at a specific timestep.

ment of a more functional web interface (integrating also a database to keep the experiments, along with some IO to be able to use MASFENON in a web environment) are planned for the future.

These tools facilitate the comprehensive interpretation of simulation outcomes and the comparison of different propagation models or scaling parameters.

3.4.3 Parameter Fitting and Integration with Real Data

MASFENON provides preliminary support for parameter fitting, enabling users to integrate real time-resolved data into the simulation pipeline. This feature is particularly valuable in domains such as biological, epidemiological, and social systems, where empirical measurements (e.g., gene expression levels, metabolite concentrations, or observed behaviors) are available at discrete time points.

The fitting process is not hard-coded into the framework core; instead, it is implemented externally through user-defined routines that update the parameters passed to the simulation engine (e.g., scaling functions for propagation, dissipation, or conservation). This design choice ensures flexibility but also means that the fitting methodology is *strongly dependent* on:

- The **type of data** available (e.g., global infection counts, node-specific

omics measurements).

- The **use case** (e.g., epidemiological dynamics vs. multi-omic cell-to-cell communication).
- The **desired outcomes** (e.g., matching overall trends vs. reproducing node-level trajectories).

For example, in the epidemiological use case, parameters are scalar, conservation dynamics are disabled, and fitting is performed on aggregated infection curves. In contrast, in the biological use case, parameters are vectorized per node, conservation is enabled, and fitting targets multi-layer, node-specific time series. These methodological differences highlight the need to adapt the fitting pipeline to the problem at hand.

Current capabilities and possible implementations include:

- Support for external optimization scripts that tune time-dependent scaling functions.
- Compatibility with common search strategies (e.g., grid search, random search, gradient-free optimizers).
- Basic scripting support for single-parameter fitting per simulation run.

Future developments will focus on:

- **Multi-parameter fitting:** simultaneous optimization of multiple parameters for a single function.
- **Graph-specific scaling:** assigning distinct fitted parameters to each agent's graph in the simulation.
- **End-to-end pipelines:** seamless integration of real-world datasets, parameter fitting, and simulation execution.
- **Generalized fitting module:** an optional framework component capable of automatically selecting and applying a fitting methodology tailored to the provided dataset and modeling goal. This module should also take into account the influence of the single parameters and their changes to the simulation, to understand the best trajectory to take in the parameter state space to find the best solution.

Ultimately, this fitting capability is envisioned as a cornerstone for connecting MASFENON simulations with experimental or observational data, enabling its use as a data-driven modeling, validation, and forecasting tool. As some example of modeling different kind of data and fitting the framework to the desired behavior and outcome, we will see some application in epidemics in section 4.2 and the most important application for this research, that is on cell to cell communication with scRNA-seq data in section 4.3.

Chapter 4

Experimental Results

This chapter presents the experimental evaluation of the MASFENON framework, providing comprehensive benchmarking results and detailed case studies. We systematically assess MASFENON’s performance and scalability within high-performance computing (HPC) environments, and demonstrate its practical utility in biological simulations, specifically cell-to-cell communication modeling. These evaluations underline MASFENON’s effectiveness and efficiency compared to existing frameworks, and highlight key insights from biological case studies.

4.1 Performance Benchmarking

To evaluate MASFENON’s computational performance, we conducted rigorous benchmarking tests. These tests were designed to assess comparative performance with established frameworks and to understand how well MASFENON scales within different HPC settings.

4.1.1 Comparison with Other Frameworks

In this subsection, we provide a performance comparison between MASFENON and two widely used agent-based modeling frameworks: NetLogo [88] and NDLib[79]. The evaluation considers three main aspects: execution time, memory usage, and scalability with respect to increasing network size. All experiments were designed to simulate epidemic spreading processes over synthetic networks of varying size, ranging from 1,000 to 30,000 nodes. We will only show the results for the range [1000,10000] since NetLogo was quite slow compared to the other two.

To ensure consistency and fairness, each framework was configured to simulate the same epidemic model (a standard SIR dynamics), under similar initial conditions. Specifically, the same initial fraction of infected nodes, infection and recovery probabilities, and network structure were used across

all frameworks. Networks were loaded from pre-generated edge list files representing undirected weighted graphs.

For NDLib, simulations were implemented using the official Python API with the "SIRModel" class, where network input was parsed via NetworkX. Memory usage and execution time were measured using GNU "time", and the performance results were averaged over multiple simulation runs.

In the case of NetLogo, the existing virus-spread model from the models library [88] was modified to support external network loading via TSV edge list, and simulations were run in headless mode via the command-line interface. Iterations were capped at 100 steps, and performance metrics were recorded similarly.

For MASFENON, simulations were executed under the same epidemic scenario, leveraging its parallel and memory-efficient multi-agent architecture. The framework's internal scheduler and propagation engine optimized the execution of epidemic dynamics while maintaining temporal consistency and reduced memory footprint. All simulations used the same synthetic network instances as the other frameworks and were repeated with matching parameters and iterations. These experiments were all done in my computer with the following hardware and software configuration:

- **Operating System:** Arch Linux (x86_64)
- **Host:** 82TD Legion 7 16IAX7
- **Kernel:** 6.15.6-arch1-1
- **Shell:** zsh 5.9
- **Screen Resolution:** 2560×1600
- **Desktop Environment:** KDE Plasma 6.4.3
- **CPU:** 12th Gen Intel Core i7-12800HX (24 cores) @ 4.70 GHz
- **GPU 1:** Intel Alder Lake-HX GT1 [UHD Graphics 770]
- **GPU 2:** NVIDIA GeForce RTX 3070 Ti Laptop GPU
- **Memory:** 64,020 MiB

To evaluate the computational performance of MASFENON, we compared it against NDLib and NetLogo in the context of epidemic simulations on networks of increasing size. Although MASFENON is designed for distributed HPC deployment, we conducted these benchmarks on a local hardware configuration to ensure a fair comparison. The reason is that NDLib and NetLogo do not natively support distributed computing environments, and running them in HPC conditions would create an unfair advantage for MASFENON due to its native parallelism.

As shown in Figures 4.1 and 4.2(also see Figures 4.3 and 4.4 for a clearer view of NDLib and MASFENON running times and memory usage), MASFENON consistently demonstrates superior performance in both execution time and memory usage, especially as the number of nodes increases. The advantages become even more pronounced in large-scale scenarios, confirming MASFENON's suitability for high-performance simulations involving large, dynamic, and interconnected networks.

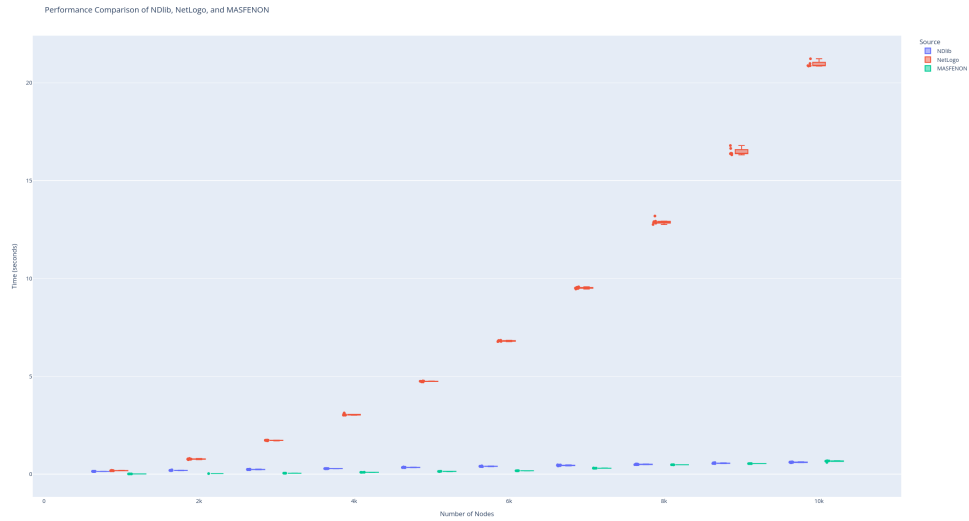


Figure 4.1: Comparison of the running times for epidemic simulations using NDLib, NetLogo, and MASFENON. Lower is better.

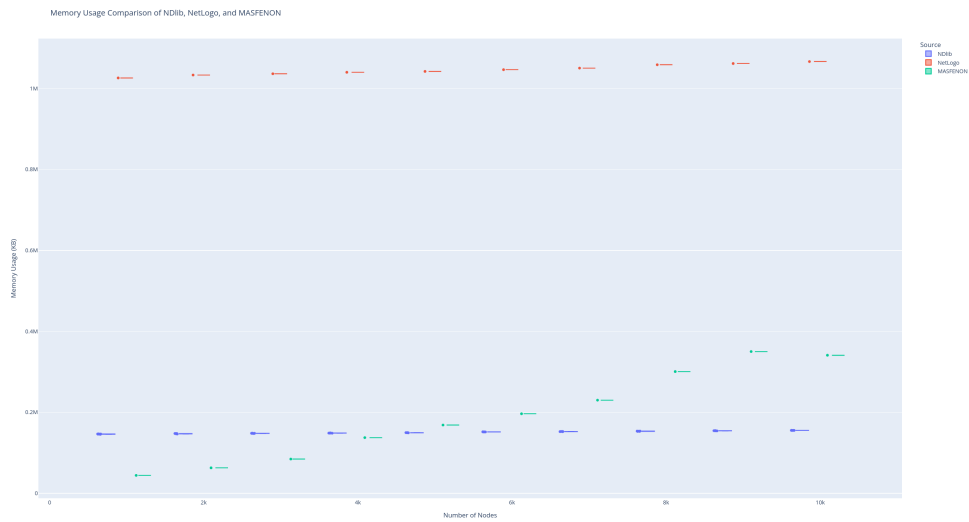


Figure 4.2: Comparison of the peak memory usage during epidemic simulations across NDLib, NetLogo, and MASFENON. Lower is better.

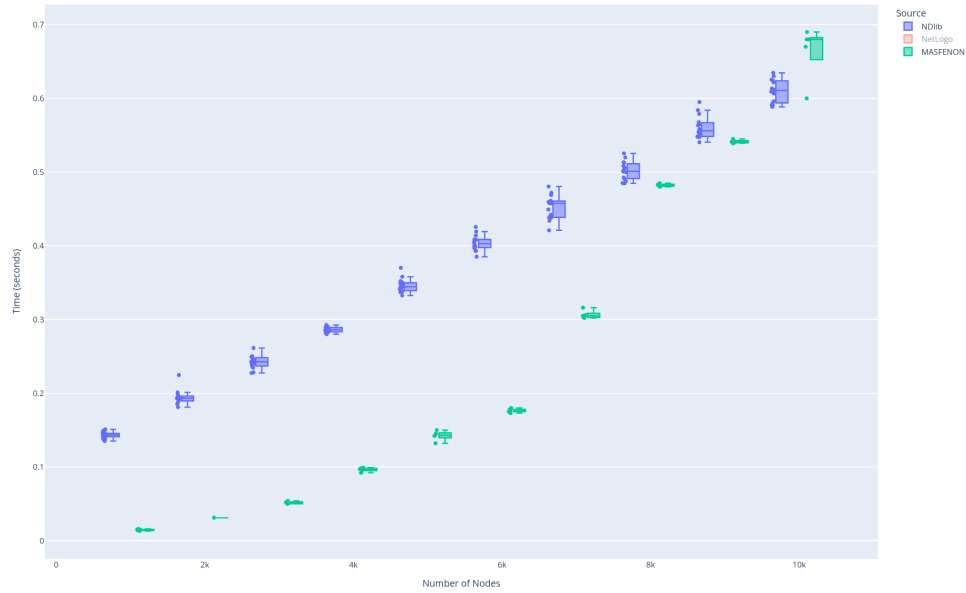


Figure 4.3: Detailed comparison of the running times for epidemic simulations using NDLib and MASFENON, without NetLogo. Lower is better.

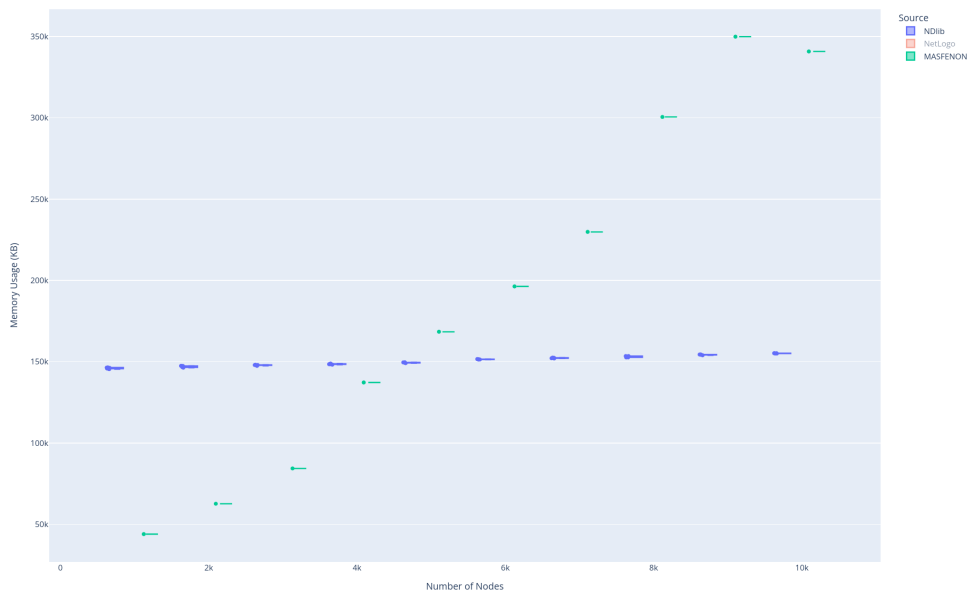


Figure 4.4: Detailed comparison of the peak memory usage during epidemic simulations across NDLib and MASFENON, without NetLogo. Lower is better.

Several key observations emerge from the results:

- **NDLib** performs efficiently for small to moderately sized networks, with both runtime and memory usage growing roughly linearly. However, it is limited in scope, it only handles compartmental epidemic simulations without extended processing (e.g., writing outputs or supporting multi-agent topologies).
- **MASFENON** not only executes the epidemic simulation but also supports additional features such as multi-agent modeling, dual-layer propagation, and the collection of simulation statistics across multiple processors. Despite these complexities, MASFENON still outperforms NDLib and NetLogo in runtime and remains competitive in memory usage.
- **NetLogo**, while more feature-rich and extensible, is significantly slower in execution and shows signs of inefficient memory handling. Despite disabling the GUI during experiments, memory usage showed irregular growth patterns, possibly due to internal data structures or delays in memory deallocation between runs.

Overall, these results validate the computational efficiency and scalability of MASFENON. Its design, centered on parallelism, memory locality, and modular execution, proves to be highly effective for simulating temporally evolving processes across large, complex networked systems.

4.1.2 Scaling Behavior in HPC Environments

Here, we specifically examine MASFENON’s scaling behavior within HPC environments, testing how effectively the framework utilizes increasing computational resources. Results from experiments across various hardware configurations and different numbers of computational nodes and processors illustrate MASFENON’s capacity for handling large-scale simulations efficiently. Performance metrics including speedup, parallel efficiency, and scalability curves are presented, showcasing the framework’s robustness and scalability.

Weak Scaling Results. Experiments evaluating weak scaling performance were conducted by increasing the number of nodes in the network while keeping the number of processors fixed. As shown in Figure 4.5a, execution time grows linearly with graph size, confirming the expected behavior due to the $O(n^3)$ complexity of matrix operations. Additional tests varying the number of communities (at fixed graph size) revealed that both very low and very high community counts tend to degrade performance (Figure 4.5b). This is attributed to suboptimal graph partitioning and communication imbalance. An optimal intermediate number of communities exists where the system is most efficient.

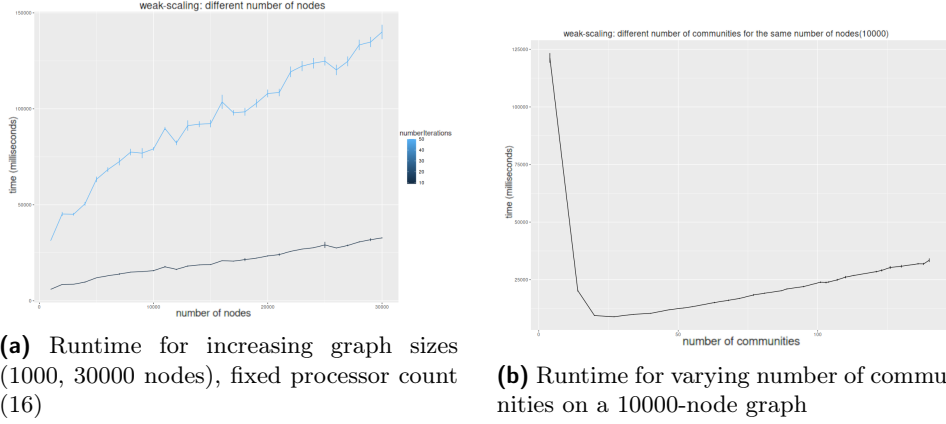


Figure 4.5: Weak scaling evaluation for MASFENON. Results show linear growth in execution time with graph size (left), and optimal runtime for intermediate community count (right). Each configuration is averaged over 5 runs; error bars indicate standard deviation.

Strong Scaling Results. Strong scaling experiments were performed by fixing graph sizes and increasing the number of MPI processors (from 1 to 32). The Barabási, Albert model was used to generate graphs ranging from 10,000 to 100,000 nodes. The experimental setup included Ice Lake CPUs (2.6 GHz), NVIDIA A100 GPUs, and high-speed Infiniband interconnects. These experiment were conducted on the Leonardo Booster supercomputer.

The results (Figure 4.6a) demonstrate a substantial decrease in execution time with increasing processor count, validating the effectiveness of parallelism in reducing runtime. The corresponding speedup (Figure 4.6b) was computed as:

$$\text{Speedup}(n) = \frac{\text{Runtime}(1P)}{\text{Runtime}(nP)},$$

showing near-linear improvement with additional processors up to saturation, especially for larger graphs.

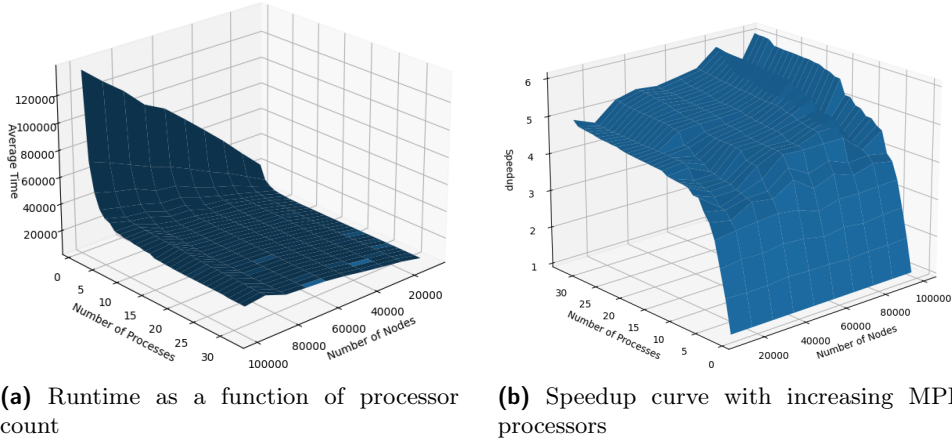


Figure 4.6: Strong scaling results for MASFENON on Barabási, Albert graphs. Each point is averaged over 5 runs. Performance gains scale with graph size and processor count.

These experiments confirm that MASFENON achieves robust weak and strong scaling performance. The hybrid parallelization strategy and communication optimizations enable simulations to scale efficiently from a few agents to thousands, across graphs with millions of interactions. This makes the framework well-suited for HPC platforms and large-scale, agent-based applications in biology, epidemiology, and complex systems modeling.

4.2 Epidemiological Simulations

Among the most compelling applications of MASFENON is the simulation of infectious disease spread within structured populations. Epidemic modeling presents a natural use case for the framework’s agent-based and network-centric architecture, allowing fine-grained simulations of intra- and inter-community transmission dynamics under different temporal and topological scenarios.

This section presents a comprehensive application of MASFENON to epidemic spreading across networks, comparing its results with classical compartmental models (SIR, SIS, SIRS), and assessing its scaling behavior on various synthetic graph structures. The full modeling pipeline, including agent initialization, parameter definition, inter-agent contact timing, and output interpretation, is demonstrated.

4.2.1 Modeling Rationale and Graphical Overview

In traditional epidemiological modeling, populations are typically represented via compartmental models where individuals are assigned to discrete categories (e.g., susceptible, infected, recovered) with transitions governed by

fixed rates. While computationally efficient, such approaches oversimplify individual variability, temporal effects, and structural properties of social contact networks, such as the presence of communities and the different spreading times of intra-community interaction VS inter-community interaction.

MASFENON departs from these assumptions by simulating epidemics on top of interconnected networks. Each agent (community) is assigned a specific graph structure and participates in temporal inter-agent communication. Propagation, dissipation, and conservation principles model the transmission dynamics within and between agents(although we will not use conservation dynamics since they do not seem to have a modeling advantage in the context of social networks and epidemics, since passing the infection does not make the infected person less infected logically). Temporal contact information governs inter-community exposure events.

Figure 4.7 summarizes the simulation workflow, highlighting: (A) the temporal network-of-networks design and virtual node communication, (B) the propagation cycle incorporating intra- and inter-agent steps, and (C) validation against statistical and epidemiological benchmarks.

4.2.2 Simulation Setup and Methodology

To formally evaluate the performance and fidelity of the framework, several epidemic experiments were conducted using well-known graph models: Erdős-Rényi, Barabási-Albert, and regular graphs. Each graph was partitioned into communities via Louvain clustering, with each community treated as a distinct agent in MASFENON.

Temporal contact events between agents were sampled uniformly and assigned to virtual node edges. Infection was initialized in 10% of the nodes, and the dynamics were simulated over 100 iterations, using:

- **Timestep:** 5
- **Intra-iterations:** $N_{\text{intra}} = 5$
- **Inter-iterations:** $N_{\text{inter}} = 20$
- **Propagation:** $\omega(t_n) = 0.5$
- **Dissipation:** $\lambda(t_n) = 0.2$
- **Saturation:** scaled tanh function with $\gamma = 1$
- **Granularity: Finer** granularity was used, to encode the singular interactions of different components of two communities.

The workflow of the epidemics modeling follows what is represented in Figure 4.8

4.2.3 Formal Validation: Statistical and Behavioral Analysis

Figure 4.9 shows the statistical distribution of infection levels at each timestep across multiple graph types and node sizes. Shapiro-Wilk normality tests

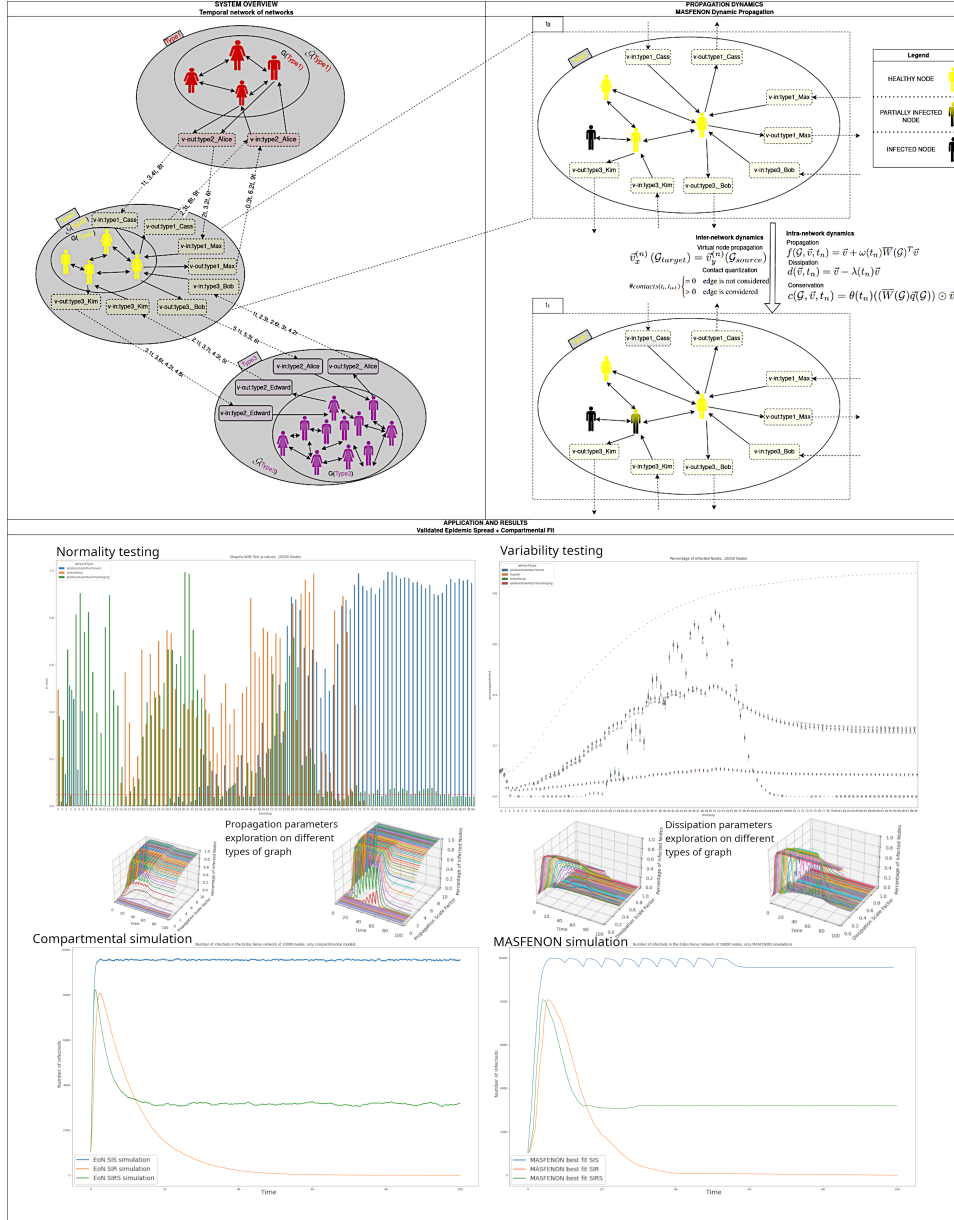


Figure 4.7: Graphical abstract of the epidemic simulation use case. **A)** System overview: agents with individual topologies and temporal virtual node connections. **B)** Propagation dynamics: intra-agent and inter-agent propagation mechanisms. **C)** Experimental validation, parameter exploration and comparison with classical epidemic models.

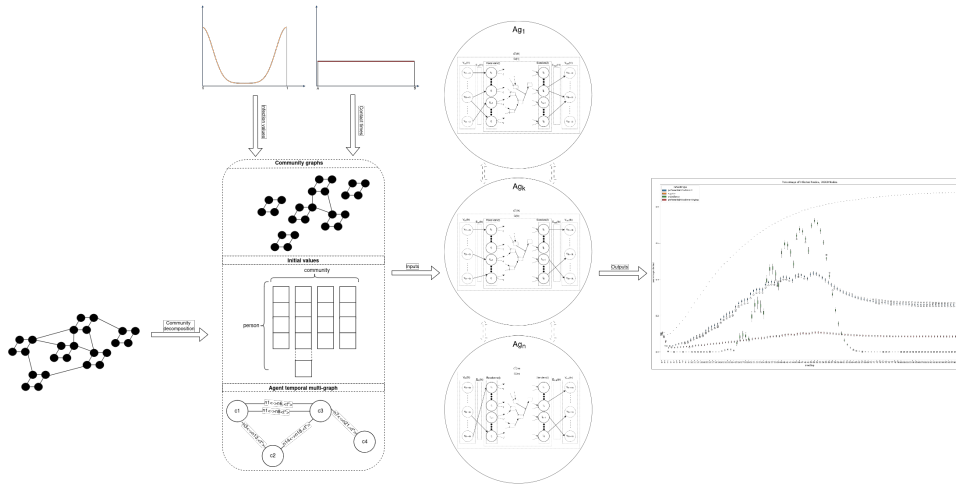


Figure 4.8: MASFENON workflow. In this example, we demonstrate a potential application of MASFENON to simulate epidemic dynamics on networks using compartmental models. The simulation begins with a network of individuals interacting through a social network. Initially, these individuals may either be infected or healthy. Next, we apply community detection techniques to partition the network into clusters, with each cluster representing a distinct community. Each community is treated as a single agent, and the interactions between these agents are modeled through inter-community connections among individuals. To introduce dynamics into the network, contact times between agents are generated randomly from a uniform distribution. This data is then fed into MASFENON, which initiates the simulation. The process requires the specification of only a few parameters. At the end of the simulation, we can analyze the temporal evolution of disease spread across the different communities.

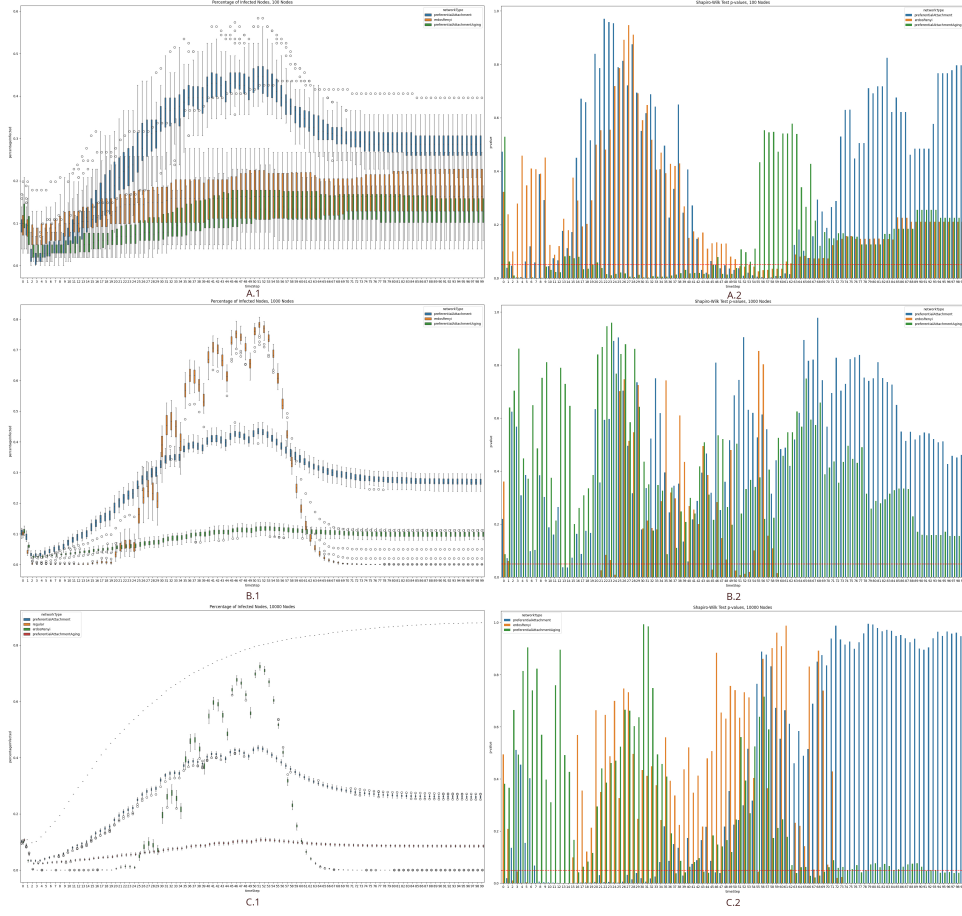


Figure 4.9: Boxplots of infection percentage (left) and Shapiro-Wilk p-values (right) across different graph models and sizes.

were applied to assess whether infection levels followed a Gaussian distribution, revealing consistently non-normal, skewed behaviors.

Key takeaways from this validation analysis include:

- Infection spread shows non-normal dynamics across graph types, as it was expected. Different infection inputs also produce different results overall (not too different when the comparison is on the same graph type, but still significant enough).
- The topology significantly affects the spreading profile.
- Communities with sparse interconnections exhibit delayed or suppressed epidemic growth.

4.2.4 Comparison with Compartmental Models

MASFENON simulations were benchmarked against standard epidemic models (SIR, SIS, SIRS), implemented via the EoN library on the same networks

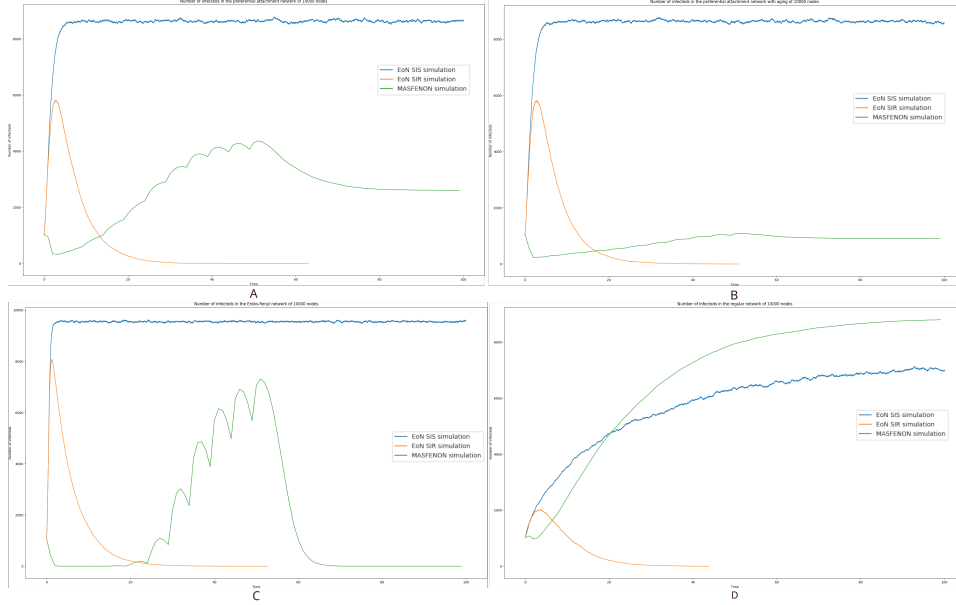


Figure 4.10: Comparison of MASFENON simulation outputs (solid) and compartmental models (dotted).

and parameters. The comparison focused on infection curves over time.

Results show that:

- MASFENON successfully emulates SIR, SIS, and SIRS behavior through appropriate tuning of $\lambda(t_n)$ and $\omega(t_n)$.
- The framework exhibits richer dynamical phenomena such as multi-stability and community-level interaction delays.
- Stepwise behaviors emerge from virtual node updates and inter-agent scheduling.

4.2.5 Parameter Sensitivity and Model Fitting

We explored the scale factor sensitivity first since we needed to see how the framework behaves in the context of epidemics starting from the propagation and dissipation scaling parameter. By adjusting parameters such as the dissipation function, propagation function, and total iterations, MASFENON can emulate SIR, SIS, and even more complex models like SIRS. For instance, tweaking dissipation and propagation scaling values can significantly modify results, as demonstrated in Fig 4.11. Dissipation affects spreading dynamics by introducing a larger decay or recovery rate, while propagation influences infection dynamics by introducing a quicker infection rate, akin to the basic reproduction number R_0 encoding the expected influence of one person in the network.

Since we want to fit MASFENON to the behavior of a real epidemic

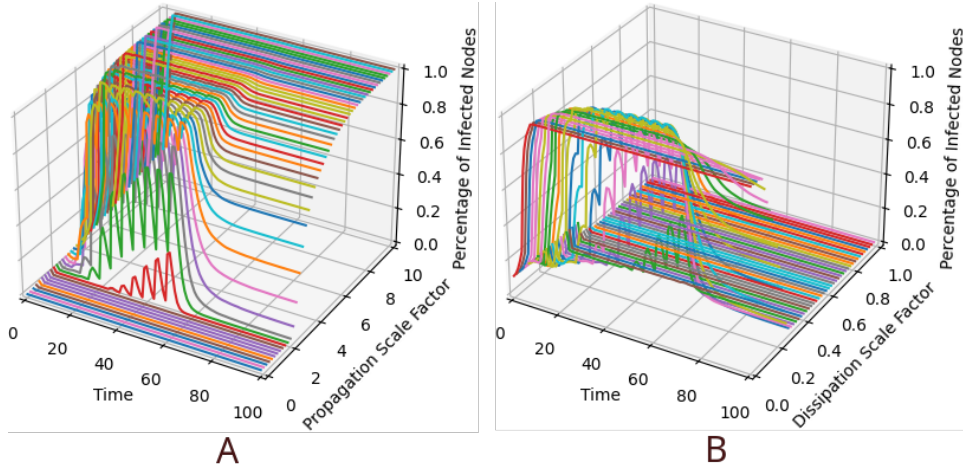


Figure 4.11: 3D plot of the Percentage of infected for the Erdos-Renyi generated graphs with 10000 nodes. These plots show the effect of the fixed scale factors for dissipation and propagation for the epidemics simulation , to show the sensitivity of the simulation to these parameters in the context of epidemics. Each line represents a specific simulation with its own parameters. A) Simulations varying the **propagation scale factor** can influence the spreading dynamics (here dissipation scale factor was fixed to 1) Simulation varying the **dissipation scale factor**(here propagation scale factor was fixed to 1)

network, we wanted to fit real data, but since real data about epidemics does not come provided with the topology of the network, we used compartmental models in place of real data, since they are already tested and validated on real data [12].

For the epidemiological simulations, MASFENON was calibrated by tuning its scalar **propagation** (ω) and **dissipation** (λ) parameters to minimize the **Residual Sum of Squares** (RSS) between the simulated infection curve and the target compartmental model trajectories (SIR, SIS, SIRS). Unlike the biological fitting case discussed in Section 4.3, the epidemiological fitting operates on *global infection counts* for the entire network rather than node-specific values and specific subnetworks or layers(while the biological case will focus on a specific layer and it's values to fit the simulation), uses *scalar* scaling parameters (constant across all nodes for a given timepoint), and does not employ the conservation term.

The fitting strategy begins with an initial guess for (λ, ω) and iteratively explores the parameter space by locally increasing or decreasing one parameter at a time, guided by the resulting RSS improvement. This process is conceptually similar to a coordinate-wise **gradient descent**, but performed in a coarse search space. The procedure is repeated sequentially across predefined time intervals: first fitting early-stage dynamics, then mid-stage, and finally late-stage behavior, until the simulated trajectory closely matches the

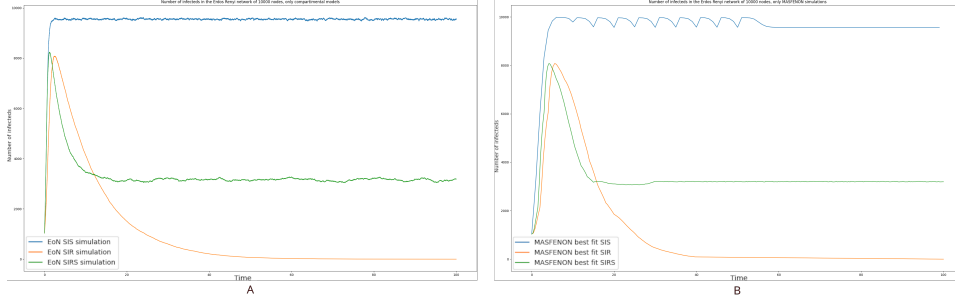


Figure 4.12: Best-fit MASFENON simulations matching SIR, SIS, and SIRS epidemic trajectories.

target compartmental model within a chosen tolerance.

All experiments were conducted on graphs with up to 10,000 nodes to ensure that the fitted parameters generalized to large-scale network structures.

The resulting best-fit dissipation and propagation functions for each model type were:

- **SIS:** constant parameters $\lambda = 0.72$, $\omega = 14.0$
- **SIR:** $\lambda(t_n) = \tanh(t_n/0.086)$, with a time-decaying $\omega(t_n)$
- **SIRS:** both $\lambda(t_n)$ and $\omega(t_n)$ varying in time

Algorithm 1 Pseudocode for Epidemiological Parameter Fitting in MASFENON

- 1: **Input:** Initial parameters λ , ω ; target curve $I_{\text{target}}(t)$; graph G ; epochs E ; learning rate η
- 2: **for** epoch = 1 to E **do**
- 3: **for** each time interval (t_i, t_{i+1}) **do**
- 4: Simulate infection curve $I_{\text{sim}}(t)$ from t_i to t_{i+1} using current (λ, ω)
- 5: Compute error: $e = I_{\text{sim}}(t_{i+1}) - I_{\text{target}}(t_{i+1})$
- 6: Update parameters:

$$\lambda \leftarrow \lambda - \eta \cdot \nabla_{\lambda} e^2$$

$$\omega \leftarrow \omega - \eta \cdot \nabla_{\omega} e^2$$

- 7: **end for**
 - 8: **end for**
 - 9: **Output:** Calibrated (λ, ω) or time-dependent $\lambda(t)$, $\omega(t)$
-

4.2.6 Scalability and Performance

Weak and strong scaling analyses were conducted using Barabási-Albert graphs up to 100,000 nodes, running on the LEONARDO supercomputer.

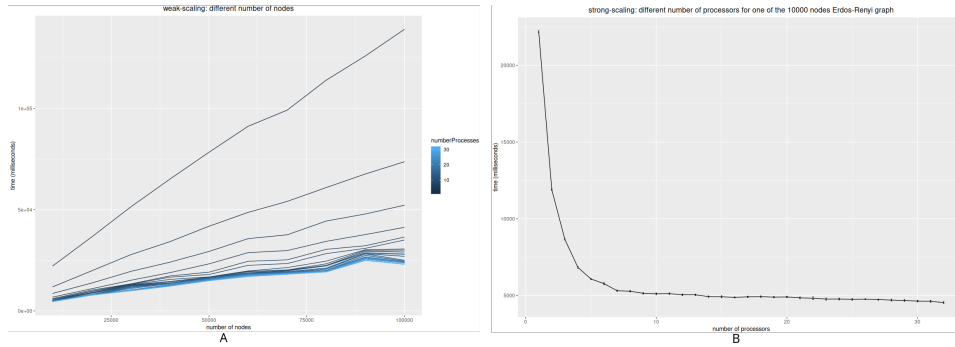


Figure 4.13: Performance results: A) execution time vs. graph size (weak scaling); B) time vs. processors (strong scaling).

Results confirm that MASFENON scales sublinearly with processor count, thanks to its hybrid MPI/OpenMP architecture.

4.2.7 Conclusions and Future Directions

MAFENON provides a powerful simulation framework for modeling epidemic spread over structured, temporally-evolving networks. It bridges the gap between simplified mathematical models and complex agent-based simulations, enabling detailed exploration of how network topology, communication frequency, and local interactions shape global outcomes.

Future directions related to the epidemiological simulation include:

- Formal integration of compartmental model equations into the propagation mechanism.
- Automated parameter fitting routines for real-world epidemic datasets.
- Support for dynamic topology changes and behavioral interventions.

These enhancements will further establish MASFENON as a general-purpose modeling tool for dynamic, interconnected systems beyond epidemiology. These future directions will be discussed in Chapter 5

4.3 Biological Simulations

As previously stated in the earlier chapters, MASFENON was conceived in the context of CCC modeling and simulation, and evolved with that use case in mind. This section is one of the most important on that point, since we will focus on the biological simulation that is one of the objective of this Ph.D. research project.

4.3.1 Introduction and Rationale

Simulating the behavior of cells within a cell-to-cell communication network is a key objective of this research. The ultimate goal is to construct a kind

of digital twin of a multicellular system, an *in silico* model that mirrors real biological interactions and dynamics. Such digital twin models have been proposed as powerful tools in biomedical research, allowing researchers to test hypotheses and predict outcomes in a virtual setting before real experiments. In complex diseases, for example, multi-cellular "virtual twin" models have been built by integrating intracellular and intercellular interaction maps[105], then converted into executable simulations that can be validated against experimental data. These models enable exploration of how cells coordinate their signals and responses, providing insights into system-level behavior that would be difficult to obtain otherwise.

Cell-cell communication in tissues drives critical processes, from development and homeostasis to disease progression, by triggering cascades of signaling pathways and metabolic changes that affect cell fate and function. Therefore, building a simulation that captures these interactions is crucial for understanding complex biological phenomena and potentially for testing therapeutic interventions *in silico*. This section describes how we assembled and simulated a multi-cellular network model of the lung cell microenvironment, leveraging single-cell data and knowledge-based networks to create a **digital twin** of cell communication in our system.

Earlier iterations of these experiments relied solely on scRNA-seq data, which still forms the foundation for the current version of the framework. At the time, the framework, then in the form of C2C-SEPIA, was focused primarily on modeling intercellular interactions based on gene expression without integrating metabolic context. The later inclusion of the metabolic network layer proved to be a critical advancement, particularly for improving the interpretability of the results and enabling the simulation of metabolite dynamics over time, an additional area of application for the framework.

Figure 4.14 shows the workflow of this earlier experimental pipeline. After initial data collection, the input scRNA-seq data undergoes preprocessing and quality control to yield normalized gene expression counts and relevant cell-specific features. Depending on the availability of metadata and annotations, earlier stages may include cell-type annotation, batch merging, or feature enrichment from external sources. This is followed by imputation, feature selection, and optionally the integration of spatial coordinates when available. The data analysis step extracts dynamic signatures (e.g., $\log_2$ fold changes) to simulate intracellular dynamics and provides cell-type assignments used for validation. The simulation stage includes intra-cellular propagation and inter-cellular communication (i.e., cell-to-cell communication or CCC), modeled respectively through intra- and inter-propagation steps. The enrichment layer handles input/output configuration and alignment to the framework's simulation engine. Finally, results are temporally analyzed, including optional spectral decomposition for dynamic pattern discovery, and cell-type-specific CCC profiles are generated.

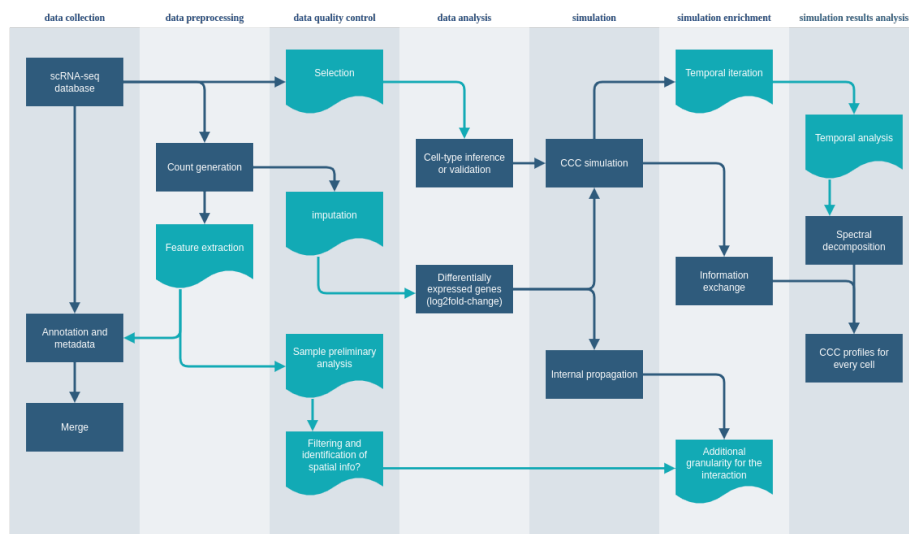


Figure 4.14: Workflow of early MASFENON experiments using only scRNA-seq data (C2C-SEPIA pipeline). The process begins with data collection, followed by optional cell-type annotation and data enrichment if such metadata is not already included. Standard preprocessing steps are applied, such as quality control, normalization, imputation, and selection of significant features. If available, spatial coordinates can also be extracted to guide interaction modeling. In the analysis stage, dynamic features (e.g., $\log_2$ fold-change) are estimated and used as simulation inputs, while annotated cell-type labels can support downstream validation. Simulation proceeds in two main phases: intra-propagation (within each cell type) and inter-propagation (modeling cell-cell communication). The enrichment step configures simulation inputs and outputs to support heterogeneous experimental setups. Final results are analyzed as temporal trajectories, with optional spectral decomposition to extract dynamic signatures. Ultimately, CCC profiles are generated for each cell type.

4.3.2 Data Sources and Preprocessing

Time-resolved scRNA-seq Data. Our simulations are based on an unpublished annotated **lung tissue** dataset generated using scRNA-seq at four timepoints: 1h, 6h, 7h, and 10h. The data contained also some other tissues and tumour data, but it was lacking the temporal data or finer resolution that was needed for the project, so we decided to stick with this data until the framework was behaving in the same way as the available data. Although genuine temporal data provides valuable biological signal, limitations include large time gaps and lack of spatial resolution. To reduce noise, we average expression values per annotated cell type, treating each cell type as an **agent** in the simulation framework.

Only three transitions are observed (1 to 6h, 6 to 7h, 7 to 10h), which may be too coarse to capture fast or non-linear dynamics in full detail. Moreover, the dataset is not spatially resolved, meaning we know the cell type of each cell and its gene expression over time, but not its physical neighbors. Despite these limitations, the dataset allows us to observe temporal trends in gene expression within individual cells and across cell types. To reduce noise and focus on broad trends, we grouped cells by their annotated cell types and averaged the expression (or downstream quantities) within each cell-type group at each time point. In our model, each cell type is treated as one *agent* or node in a higher-level network, characterized by the averaged state of that cell population at a given time. The UMAP plots for the scRNA-seq data that we are using can be seen in Figure 4.15.

For the simulation itself, we will only use the first time point to start the simulation (the 1h timepoint), while the other timepoints needs to be fitted to the simulation. The first time point data will be the initial perturbation values of the networks of each celltype (the signaling network in this case).

Metabolic Flux Inference. Because gene expression alone does not directly reveal metabolic activity, we inferred **metabolic fluxes and metabolite levels** from the scRNA-seq data using a computational tool called **scFEA** (single-cell Flux Estimation Analysis)[2]. scFEA is a graph-neural-network-based method that predicts the rate of metabolic reactions (fluxome) in single cells based on their transcriptome. In brief, scFEA leverages a curated human metabolic network (organized into modules/pathways) and applies flux-balance constraints with machine learning to estimate reaction rates from expression of enzyme-coding genes. The method captures the non-linear relationships between gene expression and reaction flux through a multi-layer model, and it was experimentally validated by showing consistency between its predicted flux changes and actual metabolomic measurements. Using scFEA (or a similar approach developed with our UK collaborators at Teesside University in Middlesbrough during my international partnership period), we obtained an estimated fluxome for each single

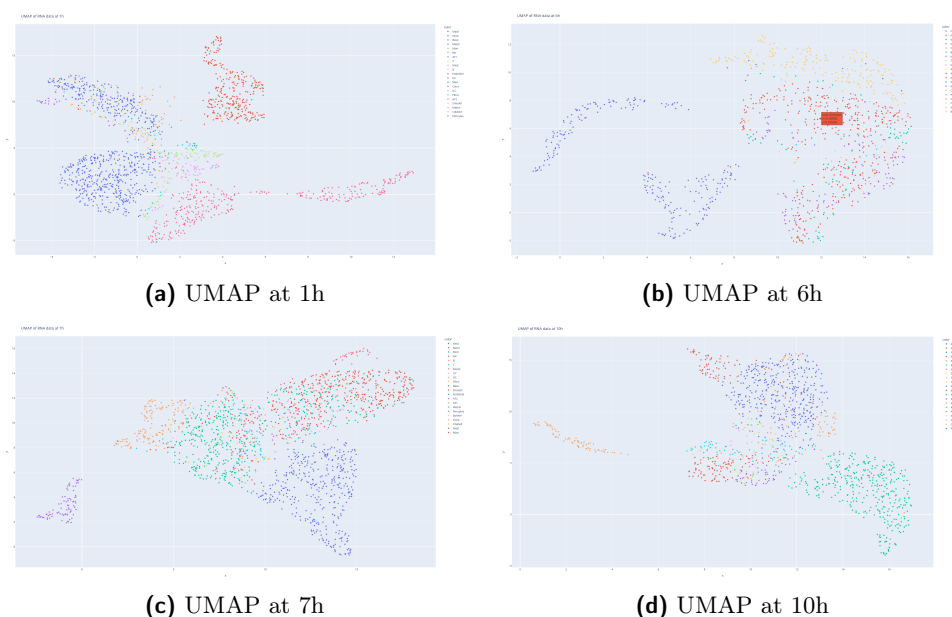


Figure 4.15: UMAP projections of the annotated single-cell RNA-seq dataset at four timepoints: 1h, 6h, 7h, and 10h. Each point corresponds to a single cell, colored by its annotated cell type. These projections highlight the temporal evolution and diversity of cell states across the experiment.

cell in the dataset. We then averaged these fluxes for cells of the same type to get a representative metabolic profile per cell type per time point. Similarly, the tool infers relative metabolite abundance changes, which we also aggregated by cell type. This processing yields time-series data not only for gene expression in each cell type but also for metabolic reaction rates and metabolite levels in each cell type, providing a multi-omic, dynamic picture of each cell type’s internal state.

In Figure 4.16, we compare metabolite inference using the graph-based machine learning method *scFEA* with the more heuristic integration of *MetalinksDB*[30] and *LIANA+*[25]. While both approaches attempt to infer metabolite-level changes from transcriptomic data, *scFEA* yields more biologically coherent outputs, especially when evaluated in the context of known pathway hierarchies and metabolic modules. That’s one of the many reasons we decided to adopt *scFEA* for the estimation of the metabolites, along the possibility of having the fluxes estimation as well (that would be used for the creation of the metabolic network edges).

We will also use *LIANA+* and *MetalinksDB* to infer the Metabolite-Receptor interactions.

Pathway Enrichment and Network Construction. Before simulation, we performed pathway perturbation analysis on the gene expression data to

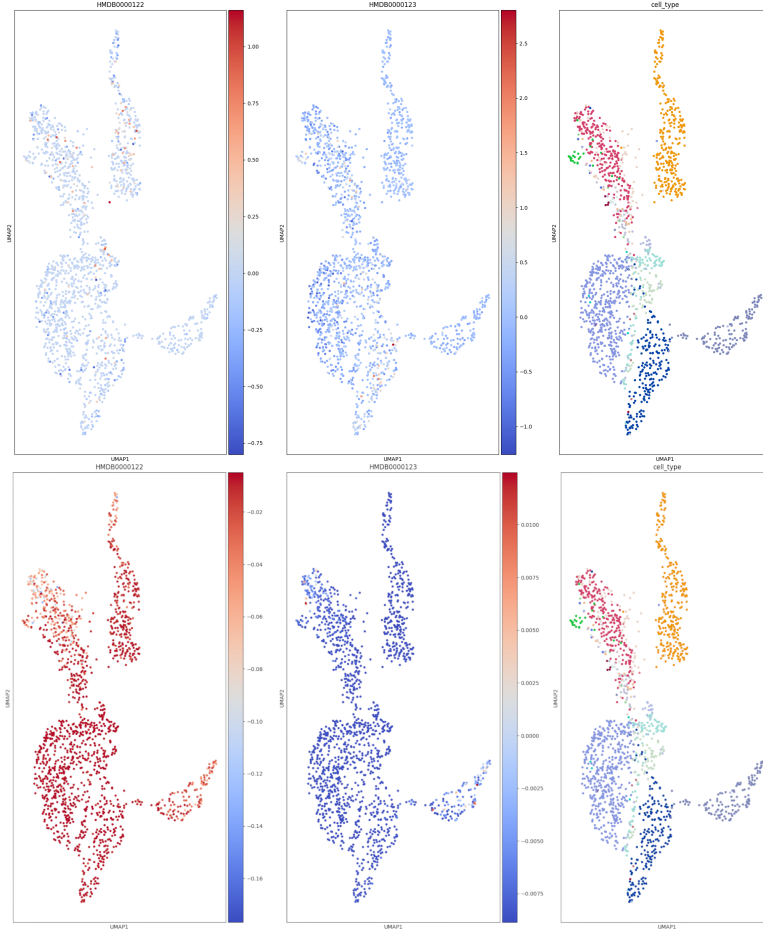


Figure 4.16: Comparison of metabolite inference using *scFEA* [2] versus *MetalinksDB* [30] and *LIANA+* [25], showing two specific metabolites (Glucose and Glycine). The inferred metabolite activity levels from *scFEA* (bottom) show higher internal consistency, pathway coherence, and resolution across cell types compared to the output from *MetalinksDB/LIANA+* (top), which exhibits noisier estimations and lacks finer resolution in certain metabolic branches. *scFEA*'s inference leverages a graph neural network trained on curated metabolic modules and integrates enzyme expression, offering improved biological plausibility for downstream simulation.

identify which signaling pathways are most dynamically perturbed in each cell type over time. This was done through differential expression and gene set enrichment analysis across timepoints, using curated pathway databases such as KEGG and Reactome. For each cell type, we selected the top enriched or perturbed pathways and merged them into a unified subnetwork, representing that cell type’s signaling layer. This method ensures that each signaling network reflects the specific biological processes most relevant to that cell type’s dynamic transcriptional behavior during the time course.

The full pipeline for constructing these signaling networks is shown in Figure 4.17. First, differential expression analysis is performed across timepoints, followed by pathway scoring and ranking. The top-ranked pathways are then expanded into interaction graphs by retrieving their molecular edges (e.g., activation, inhibition, binding) from the pathway databases. Finally, the union of these subgraphs defines a signaling-layer network for that cell type.

For metabolism, we adopt a complementary approach. Rather than customizing network structure per cell type, we begin with a global metabolic map (derived primarily from KEGG), which encodes canonical metabolite-reaction relationships. We assume that all cell types share this structure but differ in reaction fluxes, which are estimated via `scFEA` or our collaborators’ flux estimation tools. As shown in Figure 4.18, the scRNA-seq expression of enzyme-encoding genes is passed through the flux estimation pipeline to yield flux values per reaction. These values are used to weight the edges of the generic metabolic network, creating a cell-type-specific metabolic layer that encodes the likely activity of each pathway branch.

Together, the signaling and metabolic layers constitute the full intracellular interaction map for each agent (cell type), balancing data-driven dynamics with prior biological structure. This multi-layered representation captures both transcriptional signaling cascades and metabolic regulation.

4.3.3 Multi-Layer Network Architecture

Each cell type in our model is represented as an agent possessing two internal layers: a signaling layer and a metabolic layer. The signaling layer is a graph of genes/proteins and their interactions (aggregated from the perturbed KEGG/Reactome pathways), capturing regulatory and signaling processes (e.g. transcription factors, cytokine signaling cascades). The metabolic layer is a graph of metabolites and biochemical reactions (derived from KEGG metabolic pathways), capturing metabolic processes like glycolysis, TCA cycle, etc. These two layers within a cell are not isolated; they are interconnected to allow crosstalk between metabolism and gene regulation:

- **Intra-cellular Signaling → Metabolism Links:** If a gene in the signaling network encodes an enzyme or regulator that influences a metabolic reaction, we create a link from that gene node to the cor-

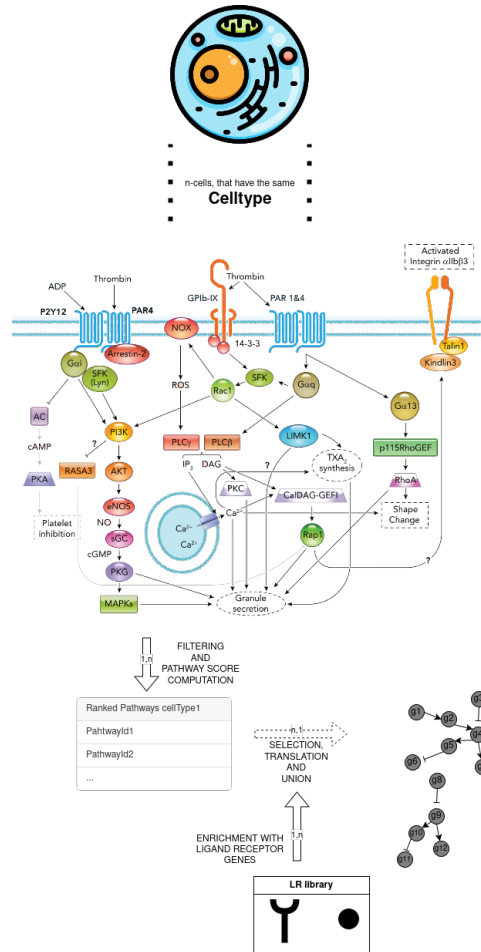


Figure 4.17: Workflow for constructing signaling pathway networks from scRNA-seq data. Gene expression data undergoes differential analysis and enrichment scoring (filtering and scoring) to identify significantly perturbed pathways at each timepoint and for each cell type (different cells with the same annotated cell type will be aggregated into the same network). For each cell type, the top pathways are extracted from pathway databases (e.g., KEGG, Reactome) and merged (selection of the important pathways, translation in case some names are not recognized, and union of these perturbed pathways) into a subnetwork that captures relevant signaling interactions. This union of pathway subgraphs becomes the cell-type-specific signaling layer used in simulation.

responding metabolic reaction in the metabolic layer. To implement this, a virtual regulatory node is introduced for each metabolic reaction, representing the reaction's activity. The gene's expression level feeds into that node, effectively modulating the reaction's rate. For example, if gene G produces an enzyme that catalyzes conversion of metabolite A to B, we add an edge from G (in the signaling layer) to the node representing the reaction $A \rightarrow B$ (or to metabolite A as the "source" of that reaction) with a weight reflecting the influence of G on that flux. This models the fact that higher expression of the enzyme gene can increase the flux of that metabolic reaction. Although we can also specify the time when these reactions happen between the metabolic and signaling layer, we choose to have them all the time (could be another point of customizability in case, since they happen all the time by default, but if they are specified they will happen at specific times). In summary, within each cell agent we augment the metabolic network with regulatory inputs from genes, thus bridging the signaling and metabolic layers so that gene expression changes can affect metabolic state.

- **Intra-cellular Metabolism $\rightarrow$ Signaling Links:** While many metabolic processes are enzyme-driven (hence captured by gene $\rightarrow$ metabolism links above), metabolites can also act as signaling molecules or have regulatory effects on gene networks. In principle, if a metabolite serves as a second messenger or allosteric regulator influencing gene expression, one could link it to the signaling layer. In our current model, we primarily consider inter-cellular metabolite signaling (described below), but within a cell we assume metabolite levels feedback mostly by well-known metabolic regulation (which is often not gene-mediated on short timescales, aside from special cases like metabolite-activated transcription factors). Thus, we did not include many direct metabolite $\rightarrow$ gene edges in the same cell, except implicitly where such pathways exist in the known signaling networks (for instance, if a pathway includes a metabolite as a ligand for a receptor in an autocrine loop).

Summarizing, each agent comprises two layers:

- **Signaling Layer:** A directed graph of genes and protein interactions, representing transcriptional and signaling regulation.
- **Metabolic Layer:** A network of biochemical reactions and metabolites, with flux weights inferred from data.

The agents (cell types) are connected in a network of inter-cellular interactions spanning both signaling and metabolic domains:

- **Ligand-Receptor Signaling (Gene $\rightarrow$ Gene between cells):** Cells communicate by secreting ligands (such as cytokines or growth factors) that bind to receptors on other cells, activating signaling pathways. We inferred such ligand, receptor (LR) interactions between each pair of cell types using the tool CellChat[43]. CellChat analyzes scRNA-

seq data for complementary expression of ligands in one cell type and their receptors in another, and computes a probability or weight for each potential interaction. From our data, CellChat produced a set of inter-cellular signaling links; for example, if cell type A highly expresses a ligand L and cell type B expresses the receptor R (and L, R is a known interacting pair), CellChat might infer a communication from $A \rightarrow B$ with a certain strength score. We incorporated these as directed edges from the ligand gene node in A’s signaling network to the receptor gene node in B’s signaling network. The edge weight corresponds to the communication strength (from CellChat’s scoring) , effectively modeling how strongly cell A’s signal influences cell B. By including these LR links, the model can propagate signaling activity from one cell type to another.

- **Metabolite-Mediated Signaling** (Metabolite $\rightarrow$ Gene between cells): Besides protein ligands, cells can also communicate through metabolites and small molecules. For instance, a cell might release lactate or ATP, which can act on receptors or sensors of neighboring cells. To capture this mode of communication, we integrated a second type of inter-cellular link: from a metabolite in one cell’s metabolic layer to a gene (or receptor) in another cell’s signaling layer. Inferring these interactions is less straightforward because the databases and methods are newer, but we leveraged LIANA+[25] (a recent framework for comprehensive cell, cell communication analysis) and the MetalinksDB[30] resource for this task. LIANA+ provides an all-in-one platform to infer cell-cell communication via diverse mediators, extending beyond classical protein ligands to include metabolites and other factors. It is built around a rich knowledge base of interactions, including metabolite, protein (metabolite, receptor) pairs. MetalinksDB, in particular, is a curated database of metabolite-protein interactions (e.g. which metabolite can bind or modulate which receptor or enzyme) aimed at inferring metabolite-mediated cell-cell communication. Using these resources, we identified candidate metabolite $\rightarrow$ gene interactions between our cell types. We then create an edge from X in A’s metabolic network to Y in B’s signaling network, weighted by a score that reflects how likely or strongly this interaction occurs under our conditions. The scoring can be informed by LIANA+ outputs or by relative expression/abundance (e.g. if cell A produces a lot of X and cell B highly expresses Y receptor, the interaction weight is high). In practice, we inferred the metabolites with another tool and we used LIANA+ and MetalinksDB to infer the interaction, while the strength of these interactions was one of the features of the inferred interactions (the weight).

By combining the above, our model is essentially a **multi-layer, multi-cell network** (a network-of-networks). Each node (cell type) contains an internal signaling network and metabolic network, and edges exist both within

nodes (signaling-to-metabolism regulatory links) and between nodes (ligand-receptor gene links and metabolite-mediated links). This forms a large integrated graph that encompasses intra-cellular dynamics and inter-cellular crosstalk.

Intra-agent Interactions. Intra-agent communication models the regulatory influence of signaling genes on metabolic activity within each cell type. Specifically, edges are introduced from signaling genes to metabolic reactions when those genes encode enzymes catalyzing the corresponding reactions. These connections are established through *virtual nodes*, which link gene expression (in the signaling layer) to the source metabolites of the corresponding reactions (in the metabolic layer). This design captures the effect of gene regulation on cellular metabolism while minimizing redundancy and ensuring structural clarity.

Although each cell type is conceptually treated as a single agent, its signaling and metabolic layers are implemented as independent processors within the simulation engine. This dual-layer abstraction enables detailed intra-cellular modeling while preserving modularity and computational efficiency at the agent level.

Figure 4.19 illustrates this mechanism: for each cell type, a predefined mapping file specifies gene-to-reaction influences, which are translated into directed intra-agent edges connecting the signaling and metabolic layers. Each influence is anchored at the source node of the metabolic module to ensure clean integration into the propagation framework.

Inter-agent Communication. Communication between agents (cell types) is modeled through two biologically grounded mechanisms, each reflecting distinct modes of cell-cell interaction:

- **Ligand-Receptor (LR) interactions**, inferred using *CellChat*[43], where edges are introduced from ligand genes in the source agent’s signaling layer to receptor genes in the target agent’s signaling layer. These directed edges are weighted based on the interaction score provided by *CellChat*.
- **Metabolite-Gene regulatory links**, inferred using *LIANA+*[25] in conjunction with *MetalinksDB*[30], where metabolic fluxes from the source agent influence gene expression in the target agent. These are modeled as directed edges from metabolites in the metabolic layer of one agent to genes in the signaling layer of another agent.

Figure 4.20 illustrates this process: we estimate LR and metabolite-receptor interactions using a multimodal combination of expression and flux data, taking into account only genes and metabolites present in the final agent-specific layers. This filtering ensures that inferred interactions are biologically plausible and structurally consistent with each cell-type’s topology.

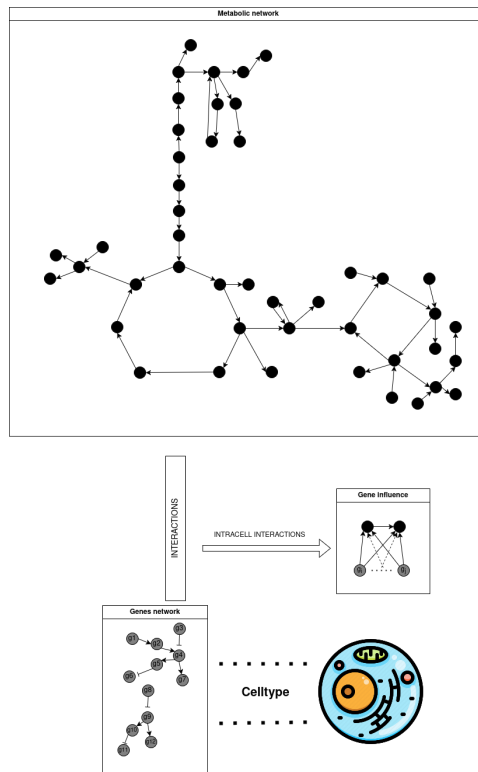


Figure 4.19: Intra-cellular interaction modeling between signaling and metabolic layers. Each influence from a signaling gene to a metabolic reaction is represented by an edge connecting the gene node to the source metabolite of the corresponding module. These links are defined through a mapping file that associates enzymes (gene products) with reactions they regulate. This design avoids redundant edges and simplifies propagation logic. In the implementation, these interactions are handled separately for each cell-type agent.

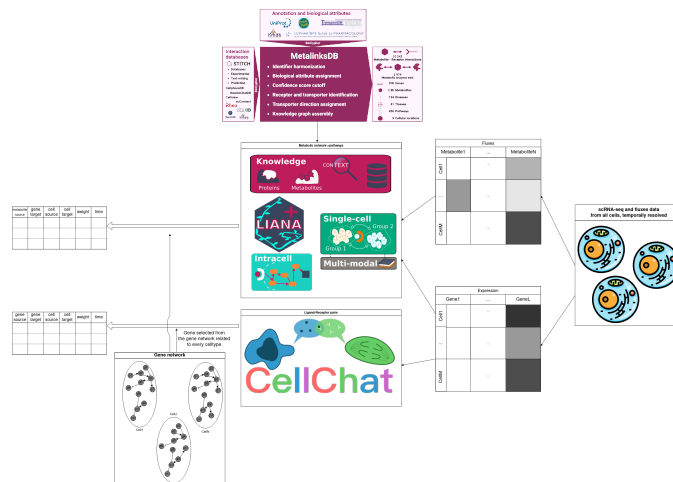


Figure 4.20: Modeling inter-agent interactions using multimodal inference. Ligand-Receptor (LR) communication between signaling layers of different cell types is inferred using **CellChat**, while metabolite-to-gene regulation across agents is derived using **LIANA+** and **MetalinksDB**, based on flux and expression matrices. Only genes and metabolites retained in the final agent-specific networks are used. The resulting LR and metabolite-receptor interactions are stored as weighted, time-resolved edge lists serving as input to the simulation.

The final outputs are two interaction lists, one for LR and one for metabolite-gene links, each containing source and target cell types, interaction strength (used as edge weight), and timepoint metadata. These lists are passed to the simulation engine as inter-agent edges in the temporal network. The granularity that was used was a **finer** granularity, to encode the different interactions of the components between two different layers and cell-types. If we want to see a general interaction between two cells, a **coarser** granularity can also be used and the output of these coarse virtual nodes can be analysed to see an aggregate response of the cells communication. This was not the target for this experiment though.

It needs to be noted that the inferred metabolites are not available for the 7h timepoint are not available, that's because the data seem to have some problems during the interaction inference (probably due to the lack of useful data and the presence of some NA values during the computation).

What we have seen before about the modeling of the data and the whole workflow pipeline can be seen in Figure 4.24 and 4.25

To summarize the complete modeling pipeline, Figure 4.25 provides an integrated overview of the full network assembly process, combining intra-agent network construction, intra-cellular crosstalk, and inter-agent communication across metabolic and signaling layers.

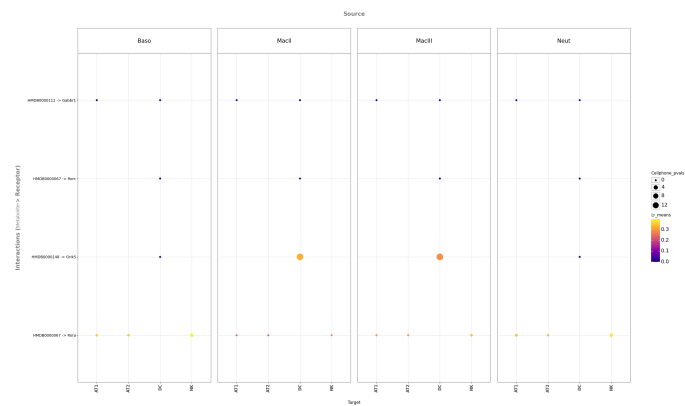


Figure 4.21: Inferred Metabolite-Receptor interactions for 1h

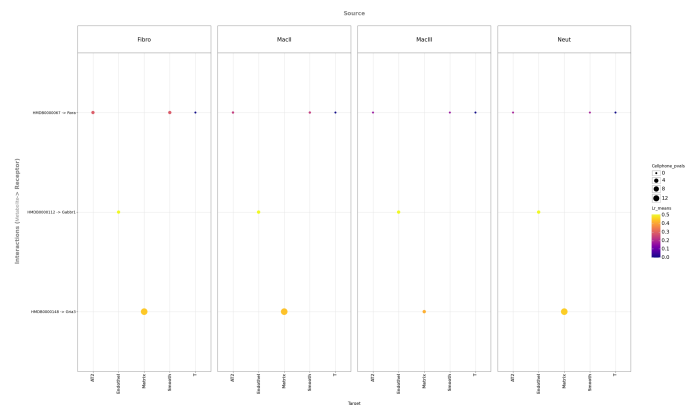


Figure 4.22: Inferred Metabolite-Receptor interactions for 6h

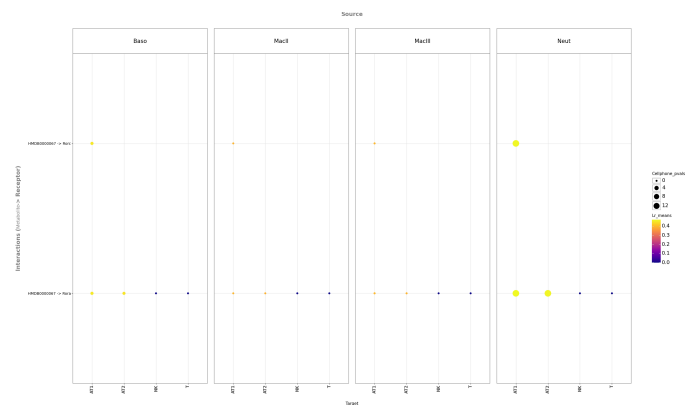


Figure 4.23: Inferred Metabolite-Receptor interactions for 10h

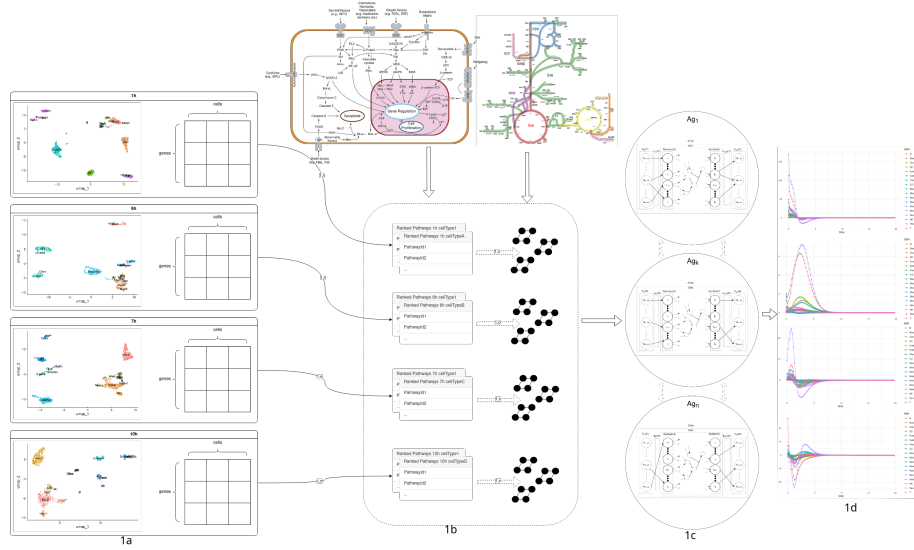


Figure 4.24: MASFENON workflow for modeling cell-to-cell communication using longitudinal scRNA-seq data. Starting from raw multi-timepoint scRNA-seq profiles and curated network resources in **1a**, the pipeline proceeds with data transformation in **1b**, where: the most perturbed pathways per (cell-type, timepoint) pair are aggregated into meta-pathways; the metabolic layer is constructed using KEGG or Reactome structure, with edge weights derived from inferred flux values; and inter-agent connections are inferred using ligand-receptor interactions and metabolite-gene interactions. In **1c**, these components are assembled into agent topologies and interaction rules, and simulation is run over time. Finally, in **1d**, temporal evolution of each cell-type's metabolic and signaling layers is visualized through simulated node-level dynamics.

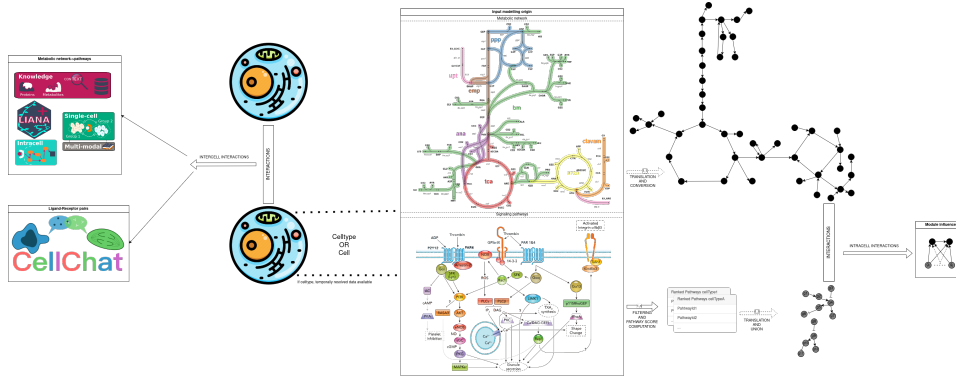


Figure 4.25: Integrated modeling of the biological simulation input for MASFENON. Each cell type (agent) is represented with two layers: a signaling network derived from the union of perturbed pathways, and a metabolic network instantiated from a common reference map with cell-type-specific fluxes. Intra-agent interactions link signaling genes to metabolic modules using virtual nodes that encode regulatory control. Inter-agent communication is modeled through ligand-receptor signaling and metabolite-gene regulation, inferred from expression and flux data. This fully connected temporal network of layered agents is passed to the simulation engine for dynamic analysis of cell-to-cell communication.

4.3.4 Simulation Setup and Parameter Fitting

With the multi-layer network constructed, we proceeded to simulate the dynamics of this system over time and to fit the model to the observed data. The simulation treats each node in the network (metabolites, genes, etc.) as having a state (e.g., concentration perturbation or activity level) that can change over time according to influences from the topology of the network and the dissipation dynamics. Because our network includes cycles and cross-talk, we employ a discrete-time iterative update approach inspired by propagation models on graphs (this process obviously mirrors the one in MASFENON since we need to have it resemble what MASFENON does to be able to simulate it). At each time step, two phases occur conceptually:

- **Intra-cellular propagation:** Within each cell's subnetwork, the states of nodes are updated based on their neighborhood nodes and on the dynamics that we define. In implementation, we propagate the values of each node in a specific network, we also apply conservation and dissipation to it.
- **Inter-cellular propagation:** After updating within nodes, we propagate signals between cells. This means taking the output of one cell's communication nodes (e.g., the amount of ligand secreted or metabolite released as computed in the intra-cell step) and transferring it to the target nodes in recipient cells. In implementation, we have "virtual output" nodes for each inter-cell edge that collect the transmitted sig-

nal, and correspondingly, "virtual input" nodes in the target cell that add this incoming influence to that cell's next update. For example, if during intra-cell propagation cell A's ligand L level increased, then in the inter-cell step, an effect is sent to cell B's receptor R node proportionally to L (scaled by the edge weight from A to B). These inter-cell updates are applied for all ligand-receptor and metabolite-gene links across the network.

By alternating these two phases over a series of discrete time steps, the model simulates how an initial perturbation spreads and evolves through the intracellular networks and intercellular communications. We ran this simulation from the initial time point (1h) through subsequent time points, effectively aiming to reproduce the trajectory observed in the data (up to 10h).

Summarizing, the simulation alternates between:

1. **Intra-agent propagation:** Updates within each cell's signaling and metabolic networks using propagation, dissipation, and conservation functions.
2. **Inter-agent propagation:** Transfer of ligand or metabolite influence through virtual nodes between agents.

Parameterization. Many parameters govern this simulation, for instance, the weights of each interaction (edge), decay rates for metabolites, activation thresholds for signaling, etc. Wherever possible, we derived these from data or literature:

- Edge weights for inter-cell links came from CellChat and MetalinksDB/LIANA+ scores as discussed.
- Initial node states at 1h were set based on the first time point measurements (e.g., the gene expression level or metabolite level in each cell type at 1h).
- Some parameters like decay or propagation functions were set to generic values or tuned to ensure stable behavior (e.g., normalizing certain influences so that values remain within a realistic range and the system doesn't diverge).
- Critically, we had to estimate reaction rate constants or influence strengths for the intra-cellular reactions and regulations. For metabolic reactions, since we inferred flux values from scRNA-seq (which are relative rates), we initialized each reaction's base rate near the inferred flux and let the gene regulatory influence modulate it. For signaling interactions (e.g., gene A $\rightarrow$ gene B activation), if quantitative data was lacking, we treated these in a Boolean/logic-inspired manner initially (e.g., gene A either sufficiently induces B, meaning that the initial weight will be 1, or not, meaning that the weight will be 0, or negatively impacts B, meaning that the weight will be -1)

Fitting to Data. The fitting procedure aimed to calibrate MASFENON so that its simulated outputs closely matched experimental observations at the available measurement timepoints. From the outset, the scaling functions for **propagation** $\vec{\omega}(t_n)$, **dissipation** $\vec{\lambda}(t_n)$, and **conservation** $\vec{\theta}(t_n)$ were treated as *time-varying, node-specific vectors*:

$$\vec{\theta}(t_n), \vec{\omega}(t_n), \vec{\lambda}(t_n) \in \mathbb{R}^{|V(\mathcal{G})|},$$

where $|V(\mathcal{G})|$ is the number of nodes in the graph $\mathcal{G}$. Each time interval $[t_n, t_{n+1}]$ thus has its own optimized scaling vectors, allowing the dynamics to vary across the simulated timeline. Conceptually, this procedure is akin to a discrete, time-local gradient descent: for each interval, parameters are updated based on the local error between the simulated and observed states at the end of that interval, and the process is repeated for a fixed number of epochs.

The dynamics, applied element-wise, are defined as:

$$c(\mathcal{G}, \vec{v}, t_n) = \vec{\theta}(t_n) \odot (\overline{W}(\mathcal{G}) \vec{q}(\mathcal{G}) \odot \vec{v}) \quad (4.1)$$

$$d(\vec{v}, t_n) = \vec{v} - \vec{\lambda}(t_n) \odot \vec{v} \quad (4.2)$$

$$f(\mathcal{G}, \vec{v}, t_n) = \vec{v} + \vec{\omega}(t_n) \odot \overline{W}(\mathcal{G})^T \vec{v} \quad (4.3)$$

Given only four experimental time points (1h, 6h, 7h, 10h), our objective was to reproduce the *trajectory shape* rather than a high-resolution time course. We therefore framed calibration as a boundary-value problem on consecutive intervals between observations. Let the observation times be

$$\mathcal{T} = \{t_1, t_2, t_3, t_4\} = \{1, 6, 7, 10\} \text{ hours},$$

Since we start from 0 in the simulation, we shift the real simulation time by 1 (i.e., we treat the 1-hour measurement as the initial input perturbation for the nodes). Define the three data-constrained intervals

$$I_1 = [t_1, t_2], \quad I_2 = (t_2, t_3], \quad I_3 = (t_3, t_4].$$

The scaling functions are modeled as piecewise-constant, node-wise and agent-associated vectors over these intervals:

$$\vec{\theta}(t) = \vec{\theta}^{(k)}, \quad \vec{\omega}(t) = \vec{\omega}^{(k)}, \quad \vec{\lambda}(t) = \vec{\lambda}^{(k)} \quad \text{for } t \in I_k, \quad k \in \{1, 2, 3\},$$

with $\vec{\theta}^{(k)}, \vec{\omega}^{(k)}, \vec{\lambda}^{(k)} \in \mathbb{R}^{|V(\mathcal{G})|}$.

Starting from the observed state at t_1 (1h), we tune the interval parameters $\{\vec{\theta}^{(1)}, \vec{\omega}^{(1)}, \vec{\lambda}^{(1)}\}$, $\{\vec{\theta}^{(2)}, \vec{\omega}^{(2)}, \vec{\lambda}^{(2)}\}$, and $\{\vec{\theta}^{(3)}, \vec{\omega}^{(3)}, \vec{\lambda}^{(3)}\}$ so that the simulated states at t_2, t_3, t_4 match the measurements while respecting biologically reasonable bounds. Concretely, when fitting the 6h point we optimize the parameters on I_1 :

$$\vec{\theta}(t) = \vec{\theta}^{(1)}, \quad \vec{\omega}(t) = \vec{\omega}^{(1)}, \quad \vec{\lambda}(t) = \vec{\lambda}^{(1)} \quad \text{for } t \in [0, 5],$$

then proceed sequentially to I_2 for 7h and I_3 for 10h. This sequential, interval-wise procedure reduces ambiguity across sparsely sampled times and keeps the fit stable while still allowing time-varying dynamics.

Incremental Fitting. To manage complexity, the fitting was performed incrementally: individual sub-networks (e.g., the metabolic layer of a single cell type) were calibrated first, then integrated into the full multi-agent system. For example, the AT1 metabolic layer was fitted to AT1-specific metabolite and flux data, adjusting parameters such as flux rates for the edges of the graph representing the metabolic layer. Parameters were initialized from experimental data (expression levels, fluxes) and iteratively refined through consecutive simulation runs, with updates constrained to biologically reasonable bounds. The fitting loop was repeated until the error function fell below a predefined threshold.

Algorithm 2 Pseudocode for Time-Dependent Vectorized Parameter Fitting in MASFENON

- 1: **Input:** Initial scaling vectors $\vec{\theta}, \vec{\omega}, \vec{\lambda}$, observed states $S_{\text{obs}}(t_i) \in \mathbb{R}^{|V|}$, graph $\mathcal{G}$, epochs E , learning rate η
- 2: **for** epoch = 1 to E **do**
- 3: **for** each consecutive timepoint pair (t_i, t_{i+1}) **do**
- 4: Simulate $\hat{S}(t_{i+1}) \in \mathbb{R}^{|V|}$ from $S_{\text{obs}}(t_i)$ using current parameters for t_i
- 5: Compute vector error: $\vec{e} = \hat{S}(t_{i+1}) - S_{\text{obs}}(t_{i+1})$
- 6: Update time-specific parameter vectors elementwise:

$$\vec{\theta}(t_i) \leftarrow \vec{\theta}(t_i) - \eta \cdot \nabla_{\vec{\theta}} \|\vec{e}\|^2$$

$$\vec{\omega}(t_i) \leftarrow \vec{\omega}(t_i) - \eta \cdot \nabla_{\vec{\omega}} \|\vec{e}\|^2$$

$$\vec{\lambda}(t_i) \leftarrow \vec{\lambda}(t_i) - \eta \cdot \nabla_{\vec{\lambda}} \|\vec{e}\|^2$$

- 7: **end for**
 - 8: **end for**
 - 9: **Output:** Calibrated parameter vectors $\vec{\theta}(t_i), \vec{\omega}(t_i), \vec{\lambda}(t_i)$
-

Limitations and Sensitivity-Based Update. While this procedure is effective, it implicitly assumes that we can compute

$$\nabla_{\vec{p}} \|\vec{e}\|^2,$$

i.e. the derivative of the vector error norm with respect to the parameters. Since the simulator is a complex, non-differentiable black-box model, exact derivatives are not accessible.

Instead, we approximate the gradient using finite displacements. For each node v ,

$$\frac{\partial \|\vec{e}\|^2}{\partial p_v} \approx 2 e_v \cdot \frac{\partial s_v}{\partial p_v},$$

where the derivative $\frac{\partial s_v}{\partial p_v}$ is estimated by the empirical sensitivity

$$\text{sensitivity}_v \approx \frac{\Delta s_v}{\Delta p_v}.$$

Sensitivity-Based Vector Update. This yields the node-wise update rule:

$$p_v^{(t+1)} = p_v^{(t)} - \eta e_v \cdot \frac{\Delta p_v}{\Delta s_v + \epsilon}, \quad \forall v \in V(\mathcal{G}), \quad (4.4)$$

which in vectorized form becomes

$$\vec{p}^{(t+1)} = \vec{p}^{(t)} - \eta \vec{e} \odot \frac{\Delta \vec{p}}{\Delta \vec{s} + \epsilon}, \quad (4.5)$$

with $\odot$ denoting elementwise multiplication and the fraction taken element-wise.

It is worth noting that, although sensitivity is defined as $\frac{\Delta s}{\Delta p}$, the update rule employs its reciprocal form $\frac{\Delta p}{\Delta s}$. The reason is that sensitivity quantifies how the simulation output changes in response to parameter variations, while in the fitting procedure we need the inverse relation: how much the parameter must change in order to correct a given discrepancy in the output. In other words, whereas $\frac{\Delta s}{\Delta p}$ measures the forward effect of parameters on the simulation, the update step requires estimating the backward adjustment of parameters given an error in the simulation. This inversion ensures that the parameter correction is scaled consistently with the observed responsiveness of the system.

More formally, we defined the sensitivity as:

$$\text{sensitivity} \approx \frac{\Delta s}{\Delta p},$$

which is a finite-difference approximation of the derivative

$$\frac{\partial s}{\partial p}.$$

Suppose our error is

$$e = s - s^{\text{real}}, \quad \|e\|^2 = (s - s^{\text{real}})^2.$$

Then

$$\frac{\partial}{\partial p} \|e\|^2 = 2e \frac{\partial s}{\partial p}.$$

So a true gradient descent update would be:

$$p^{(t+1)} = p^{(t)} - \eta \cdot \left(2e \frac{\partial s}{\partial p} \right).$$

In our **finite-difference formulation**, we are not directly computing $\frac{\partial s}{\partial p}$, but want to **estimate how much to move p to reduce e** .

Rearranging:

$$\Delta p \approx \frac{\Delta s}{\frac{\Delta s}{\Delta p}}.$$

So if we want a correction proportional to the needed change in p to offset an error e in s , we flip the ratio:

$$\Delta p_{\text{corr}} \sim \frac{e}{\frac{\Delta s}{\Delta p}} = e \cdot \frac{\Delta p}{\Delta s}.$$

That is exactly where the inverse comes from: instead of asking **how much does s change when p changes** (the sensitivity), we want to know **how much should p change to correct a change in s** .

Meaning that the formula seen in equation is **not using sensitivity directly**, but rather its reciprocal, because you want to solve for the required Δp given an error in s .

Approximate Independence Assumption. This elementwise update implicitly assumes that node-wise errors are approximately independent under small perturbations. In reality, propagation within the graph couples the dynamics, meaning that $\vec{e}$ has correlated entries. Nevertheless, this approximation provides a tractable fitting procedure, as analyzing the full coupled dynamics would be prohibitively complex.

Interpretation. This discretized formulation ensures that parameter corrections respect the observed effect of parameter displacements on simulation outcomes. If increasing p_v leads to an increase in s_v ($\Delta s_v / \Delta p_v > 0$), a positive error implies reducing that parameter; conversely, if $\Delta s_v / \Delta p_v < 0$, the correction naturally reverses direction. The result is a fitting procedure that is both derivative-aware and biologically consistent, while remaining computationally feasible for large, coupled multi-layer networks.

Example: AT1 Metabolic Network Fitting. AT1 cells (alveolar type I pneumocytes) are specialized epithelial cells responsible for gas exchange and characterized by a distinct metabolic program. For this example, we extracted the subgraph of the AT1 **metabolic layer** corresponding to key pathways relevant to its physiology (e.g., glycolysis, TCA cycle, surfactant metabolism) and performed a fitting procedure to match the simulated

metabolite trajectories to the measured time series of AT1 metabolite abundances. We still simulated the whole network of cells that are available for the data, but we only focused the fitting process for the AT1 metabolites.

The fitted model was able to qualitatively reproduce well-known biological patterns, such as a wave of signaling activation following an initial stimulus and a subsequent metabolic shift in selected cell types. Nevertheless, discrepancies remained, particularly in cell types with noisy or sparse metabolite measurements, highlighting the need for either additional data or further refinement of the model.

Figure 4.26 shows a subset of metabolites from the AT1 metabolic network after fitting, including *beta-Alanine*, *Uridine triphosphate*, *Serine*, *Dolichyl phosphate D-mannose*, *Choline*, *Leucine*, *Uridine diphosphate-N-acetylglucosamine*, *Malic Acid*, and *alpha-D-Glucose*. These are shown as a proof of concept rather than an exhaustive display; the network also contains other metabolites and virtual nodes. We place particular emphasis on *alpha-D-Glucose* due to its biological importance and its relevance in ongoing research projects.

It is important to note that some metabolites in the simulation lack corresponding real measurements. This is because **scFEA** estimates approximately 70 metabolites, whereas our curated AT1 metabolic network contains about 102 metabolites (excluding virtual nodes; the augmented network exceeds 200 nodes in total). Despite this mismatch, the fitted simulations capture the general trends of the available experimental data, though not always the exact values. For instance, the real *alpha-D-Glucose* profile exhibits an unusual pattern, increasing from 1h to 6h, then dropping sharply, making it challenging to replicate given the constraints of the network topology and propagation dynamics. Similar erratic patterns appear for other metabolites, yet the simulation is still able to approximate the direction of changes and, in some cases, match specific time points almost perfectly. Overall, these results demonstrate that MASFENON can effectively fit complex, noisy biological data and reproduce key dynamic trends in a realistic cellular network.

4.3.5 Discussion and Limitations

The biological simulation described above demonstrates a framework to **digitally emulate cell-to-cell communication** using real data. By integrating scRNA-seq, inferred metabolic fluxes, and pathway knowledge, we created a multi-cell network model (effectively a prototype digital twin of the lung cell microenvironment) and showed that it can be tuned to reflect observed temporal changes. This approach is ambitious and comes with several limitations:

- **Data Limitations:** The time-series scRNA-seq dataset had only four time points with a wide gap between the first and later times. This makes it challenging to capture fast dynamics or to distinguish di-



Figure 4.26: Plots of the MASFENON simulation results for some metabolites after the fitting process.

rect effects from indirect ones. The model can interpolate between time points, but without finer sampling we might miss oscillatory or transient behavior. Moreover, averaging over cell populations (while necessary to reduce noise) removes cell-to-cell variability, potentially masking important heterogeneity in responses. In the future, more densely sampled time points or single-cell trajectory methods could improve the model’s temporal resolution. Incorporating spatial information (e.g., which cell types are adjacent in tissue) would also refine inter-cellular link predictions beyond what expression alone can tell. We have also observed in Figure 4.26 that the real metabolite data display irregular fluctuations between consecutive timepoints. While it is not possible to determine with certainty whether this represents genuine biological dynamics or technical variability without further experimental validation, such abrupt variations appear atypical given the expected continuity of metabolic processes, which usually follow smoother trajectories of production and consumption over time. Possible explanations include measurement noise, normalization artifacts, or transient metabolic responses that were not fully captured by the limited temporal resolution of the dataset. This observation highlights the importance of incorporating robust fitting and smoothing mechanisms in MASFENON when dealing with biological time-series data, to better disentangle experimental noise from meaningful dynamic behavior.

- **Model Complexity and Uncertainty:** The integrated network contains many nodes and interactions, making it a high-dimensional system. There is uncertainty in the structure (did we include all relevant pathways? are all inferred LR interactions true positives?) and in the parameters (exact reaction kinetics, etc.). We addressed some uncertainty by focusing on the "most perturbed" pathways and using databases for known interactions, but some interactions in the model may not actually be active in our specific biological context. The parameter fitting helps validate some interactions (those necessary to fit the data), but others might be theoretically present but practically redundant. Overfitting is also a risk given the small number of data points, we mitigated this by keeping parameter values within biologically plausible ranges and by using independent data (like literature-known behavior) as a sanity check. Still, the model should be interpreted as one plausible instantiation of the system dynamics rather than a definitive ground truth.
- **Simulation Approach:** We used a custom iterative propagation algorithm (inspired by frameworks like MASFENON and others) rather than classical differential equation solvers. This was suitable for a qualitative fit and allowed flexibility (e.g., easily incorporating logic-like rules and thresholds). However, it introduces its own parameters (such as the number of iterations per hour, the functional form of propagation and dissipation, etc.). Tuning these was somewhat empirical. An alternative approach could be to formulate the model entirely as a system of ODEs or a Boolean network. Indeed, others have built large-scale Boolean models for multi-cellular systems and succeeded in validating them against steady-state data. A Boolean model would simplify parameter estimation (no kinetic constants required) but at the cost of quantitative prediction (only on/off states). We chose a semi-quantitative approach to leverage the continuous nature of the scRNA-seq and flux data. In future work, exploring hybrid modeling (continuous for metabolic, discrete for certain signaling) could be beneficial.
- **Inter-cellular Interaction Inference:** Tools like CellChat and LIANA+ greatly aided the network construction by suggesting likely cell-cell communications. CellChat has been well validated to infer signaling networks from scRNA-seq, but it is not perfect, it may miss interactions that involve post-transcriptional regulation or over-predict interactions when cell types have common ligand/receptor expression. Likewise, metabolite-mediated interactions from MetalinksDB are a new frontier; many are hypothetical or context-dependent. We noticed that using the raw LIANA+ pipeline yielded some biologically implausible metabolite links (likely false positives), which is why we involved expert curation. As metabolomics and multi-omics data become more

integrated, these predictions will improve. For now, our model likely under-represents metabolite signaling (since we favored precision over recall in including only confident links).

- **Fitting Process:** The current fitting strategy is based on minimizing an error function over the **aggregate** simulation output on a single target layer of interest (AT1 metabolite layer in this research). This approach overlooks general deviations at the node level for other layers and may obscure biologically significant discrepancies. As a result, it limits the precision of validation and the interpretability of fine-grained behaviors within the system. Addressing this issue will be a priority in future iterations of the framework, as discussed in Chapter 5.

Nonetheless, the resulting model provides a proof of concept as a functional *digital twin* of the lung cell environment. It captures intra- and inter-cellular dynamics with biological realism and lays the foundation for subsequent simulation of perturbations, interventions, and disease scenarios.

In conclusion, this **biological simulation** chapter has detailed the construction of a computational model that mirrors cell-to-cell communication networks using real single-cell data. We demonstrated the model on a specific use case (metabolic network of AT1 cells) and showed that it can be calibrated to experimental observations. This lays the groundwork for a full digital twin of the lung cell environment, which in subsequent sections (e.g., theoretical use cases 6 and future work 5) we will use to explore "what-if" scenarios. The hope is that such a model can provide insights into how cellular interactions produce emergent behaviors, and ultimately guide experimental design by allowing in silico experimentation, much as early studies have suggested in other contexts

Future sections will explore theoretical use cases of this model and potential extensions to fully dynamic, real-time virtual organs.

Chapter 5

Future Work and Extensions

The experimental results presented in Chapter 4 demonstrate the flexibility, scalability, and modeling capabilities of MASFENON across diverse use cases, from epidemiological simulations to biological modeling of cell-to-cell communication. At the same time, these results highlighted several limitations and challenges that point to promising avenues for future development.

Some of these directions are motivated by practical constraints encountered during fitting procedures (e.g., data sparsity, limited number of time points, coarse-grained fitting targets, overfitting and validation), while others emerge from performance profiling and the need for greater modeling expressiveness or adaptability, born from the need for some data types (like spatial transcriptomics).

5.1 Enhancements for General Use and Flexibility

Future developments could focus on increasing the framework’s adaptability to different network and agent models:

- **Intra-temporal dynamics:** Extend the framework to support temporal changes within each agent’s internal network, enabling simulations where topology evolves over time within agents, addressing limitations noted in modeling static intra-agent layers.
- **Hypergraph support:** Introduce native support for hypergraphs to capture multi-way interactions that cannot be reduced to pairwise edges without information loss, which would be beneficial for modeling complex multi-participant reactions or multi-node communication events. This support could be a key feature for multi-node interactions, such as multi-subunit ligand-receptor assemblies, protein complexes, or cooperative signaling cascades.
- **Dynamic code loading:** Explore **reflection mechanisms** or dynamically loaded libraries to allow runtime definition of new propagation, dissipation, and conservation functions without recompilation.

- **Custom linear algebra backend:** Move beyond the Armadillo dependency by developing a dedicated linear algebra library optimized for MASFENON's specific access patterns, potentially improving performance bottlenecks identified in large-scale scaling experiments.
- **Vectorized node states:** Generalize the framework so that each node state is represented as a vector or tensor, allowing richer state variables to be propagated (e.g., multi-omics data in biological models, spatial transcriptomics or multiple states in social simulations).
- **Topology refinement:** Implement procedures for fitting and refining input network topologies themselves, enabling the model to adapt its structural assumptions to better match real data, a feature that would directly address limitations seen in chapter 4.

5.2 Enhancements Specific to Biological Modeling

The current biological modeling capabilities (Section 4.3) could be expanded to capture greater complexity and realism:

- Incorporate domain-specific biological constraints, such as reaction kinetics, compartmentalization, or stoichiometric balance, to make metabolic and signaling simulations more biologically plausible.
- Enable multi-omics integration, where transcriptomic, proteomic, and metabolomic layers can be modeled simultaneously, reducing the reliance on inferred fluxes alone.
- Support parameter inference pipelines tailored to biological datasets, including uncertainty quantification for fitted propagation, dissipation, and conservation functions, helping to mitigate overfitting risks observed in sparse datasets.

5.3 Integration with Real-World Biological Data

A key extension is the seamless integration of MASFENON with large-scale biological datasets:

- Develop preprocessing pipelines for widely used formats in single-cell and spatial transcriptomics, such as `.h5ad` and `.loom`, reducing the manual preprocessing described in Section 4.3.
- Implement data-driven initialization of both node states and network topology from raw measurements.
- Automate the fitting process for heterogeneous datasets, potentially leveraging machine learning for parameter optimization and for refining inter-agent and intra-agent structures simultaneously. This process is also not only related to real-world biological data, but it should be used with any kind of data. as long as the modeling is done in the correct way.

- Have some validation modules and scripts implemented, to be able to address the overfitting and have a more stable fitting process.

5.4 Potential Optimizations for Exascale Computing

The performance results in Section 4.1 show strong scaling, but also highlight areas for optimization, especially for very large topologies:

- **GPU acceleration:** Implement CUDA/HIP kernels for propagation, dissipation, and conservation to fully exploit GPU resources.
- **Hybrid CPU-GPU execution:** Develop a scheduler for efficient workload distribution across CPUs and GPUs, minimizing communication bottlenecks observed in high-rank MPI runs.
- **Profiling-driven optimization:** Integrate detailed profiling to guide code refactoring and to quantify the cost of core operations.

We expect a speedup of 1.5 when fully optimized, since there are still parts of the code dependent on external libraries that are not yet fully optimized for a HPC setting (such as Armadillo), meaning that almost all the linear algebra can be successfully optimized.

5.5 User Interaction and Visualization

Improving accessibility and interpretability is essential for broader adoption:

- Create a web-based interface for simulation configuration, execution, and monitoring, replacing ad-hoc scripts currently needed.
- Enable real-time visualization of distributed simulations and streaming of intermediate results.
- Extend visualization capabilities for temporal multi-layer networks, supporting animated exploration of both intra-agent and inter-agent dynamics.

In summary, the next steps for MASFENON will focus on making the framework more adaptable to diverse use cases, more biologically realistic, and more efficient at extreme scales, while also improving its usability for non-expert users. Many of these enhancements directly address the limitations and bottlenecks identified in the experimental results, ensuring a continuous feedback loop between performance evaluation and framework evolution.

Chapter 6

Use Cases and Applications

MASFENON’s versatility positions it as a powerful tool for simulating and analyzing dynamics within various complex systems. Its ability to represent detailed temporally and spatially resolved interactions makes it suitable for a broad range of applications, from biological systems to epidemiological models and beyond.

This chapter presents a selection of representative use cases, illustrating how MASFENON can be adapted to diverse domains. These examples build upon the results in Chapter 4 and emphasize both already implemented applications and potential future extensions.

6.1 Simulating Cell-To-Cell communication

Cell-to-cell communication (CCC) underpins essential biological processes, driving development, immune responses, and pathological transitions such as cancer progression. Traditional modeling approaches often lack either the granularity or the temporal resolution required to fully capture these dynamics.

As demonstrated in Section 4.3, MASFENON can simulate CCC using scRNA-seq data. The same framework can be extended to incorporate additional modalities, such as **spatial transcriptomics**, by leveraging its modular design. In particular, the use of intra- and inter-agent propagation, virtual nodes, and vectorized scaling parameters enables a flexible representation of spatially constrained cellular interactions. Through these features, MASFENON supports:

- Identification of signaling patterns between spatially localized cell types.
- Exploration of metabolic-signaling interplay both within and across tissues.
- Realistic simulations that integrate spatial context into data-driven models.

In the case of **spatial transcriptomics**, bulk sequencing data is enriched

with spatial information, opening multiple levels of abstraction for simulation design:

1. **Single-cell level.** After deconvolution of spatial spots into single-cell profiles, agents can be defined as individual cells with both type and spatial coordinates. Interactions between cells incorporate not only their type-specific ligand-receptor relations but also distance-dependent weights, capturing locality effects.
2. **Spot level.** Each spatial spot is modeled as a single agent. Interactions are defined through differentially expressed ligand-receptor pairs across spots, while the neighborhood structure of the spatial array constrains interaction strength. Distant spots may be pruned to enforce biologically realistic locality.
3. **Tissue/region level.** Areas of the spatial plane, classified by shared type or tissue identity, are abstracted as agents. Interactions occur at the boundaries between tissues and are selected from the set of valid CCI interactions, enabling higher-level modeling of tissue-tissue communication.

Looking ahead, future extensions of MASFENON (see Chapter 5) will include the ability to represent *vector-valued node attributes*, beyond the vectorized scaling functions already available. This will allow nodes to represent entire molecular profiles (e.g., multi-gene expression vectors), enabling a higher-order representation of spatial agents where intra-agent dynamics operate directly on high-dimensional states.

In summary, spatial transcriptomics provides a natural extension for CCC simulations in MASFENON, offering multiple granularities of modeling, from fine single-cell resolution to coarse tissue-level abstractions, all unified under the same simulation principles.

6.2 Epidemic Spread Modeling

Epidemiological modeling is crucial for public health policy and outbreak response. MASFENON provides detailed simulations of infectious disease spread, accounting for heterogeneous contact structures, temporal variability, and intervention strategies.

Building on the experiments in Section 4.2, the framework can:

- Simulate outbreak trajectories on large, realistic contact networks.
- Compare results with standard compartmental models (SIR, SIS, SIRS).
- Incorporate dynamic changes in interaction patterns over time.

Its high scalability makes it suitable for real-time scenario exploration during emerging health crises.

6.3 Social Network Diffusion

Information diffusion in social networks shares many characteristics with epidemic spread. However, the transmitted "signal" is often abstract (e.g., influence, opinion, or visibility) rather than a biological pathogen.

Following the epidemiological modeling approach, MASFENON can simulate social diffusion processes at various levels of granularity. As an example, we consider the **propagation of scholarly influence** in academic networks.

6.3.1 Motivation and Context

Scholarly impact evolves through networks of co-authorship, citations, and topical relevance. Traditional bibliometric analyses focus on static measures (e.g., h-index, total citations), neglecting the temporal and relational dynamics of influence propagation.

MAFENON enables dynamic modeling by treating research groups, institutions, or individual researchers as agents with internal publication networks, linked via collaborative and citation-based interactions.

6.3.2 Model Design and Simulation Setup

In this use case:

- **Agents** represent research groups, topics, or individuals.
- **Nodes** in an agent's internal graph correspond to publications.
- **Edges** represent co-authorship links, citation flows, or topic overlap.
- **Node values** encode publication influence or citation potential.
- **Inter-agent communication** models collaborations and cross-citations using virtual nodes and timestamped interactions.

Propagation functions capture influence spread within an agent's publication network, while dissipation functions model the decay of attention to older works.

Inter-agent communication captures academic exchanges such as joint authorship, group cooperation, different topic cooperations or institutional collaborations. These are encoded using virtual nodes and timestamped contact events, just as in the epidemic use case. The type of granularity for these inter-agent interactions can be **coarser** (if we want to encode the amount of information two different agents pass between each other, such as the relation between two research groups or two different topics), or **finer** (to encode the intra-agent dynamics evolution by also considering the relationship of nodes from different networks).

Simulation parameters include:

- **Initial state vector:** Publication scores (e.g., normalized citation counts) for each paper node.

- **Propagation function:** Models internal influence of highly cited papers on newer work within the same agent.
- **Dissipation function:** Encodes decay of attention or relevance of older work over time.
- **Conservation function:** Represents cumulative reputation retention or author-specific weights. Could also not be considered as in the epidemics case, since the passing of information between two nodes does not mean that the information is lost on the node that is passing the information.
- **Virtual node communication:** Transfers scholarly signals between agents based on shared papers or citations.

6.3.3 Data and Implementation

Data sources could include:

- Co-authorship networks (e.g., from DBLP or Scopus).
- Citation graphs and bibliometric scores.
- Topic modeling outputs for research field classification.

Agents can be defined through community detection on co-authorship networks, or by predefined topic or institutional boundaries.

6.3.4 Simulation Insights

Simulation results could reflect how influence concentrates in high-impact clusters, or how inter-agent communication (e.g., co-authorship across domains and communities) can significantly amplify h-index-like metrics. hypothesized key patterns that could be observed could include:

- **Localized propagation:** High-impact publications elevate other works in the same community through repeated propagation steps.
- **Cross-community reinforcement:** Joint publications or citations enable mutual reinforcement of scholarly visibility.
- **Temporal evolution:** Shifts in influence across time are visible, with emerging communities gaining or losing prominence depending on collaboration and novelty.

6.3.5 Discussion and Use Case Potential

This sociological use case highlights how MASFENON can simulate complex temporal phenomena beyond classical physical or biological processes. By reinterpreting node values as abstract quantities like "influence" or "visibility," and by structuring agent interactions through academic networks, the framework could enable fine-grained, time-aware simulations of scholarly ecosystems. This type of process can be seen as a type of epidemiological case where there are no fuzzy states, but the values are the influence of the node over all the networks (the author, the paper or the topic of interest).

In future developments, this module could:

- Support real-world h-index forecasting based on historical publication records.
- Integrate bibliometric databases (e.g., Scopus, DBLP, Semantic Scholar) to simulate research impact evolution.
- Offer predictive insights for funding agencies or policy makers on collaboration strategies.

This use case solidifies the general-purpose adaptability of MASFENON and opens the door to its application in other domains such as social network analysis, knowledge diffusion, or innovation modeling.

6.4 Economics and Organizational Networks

Economic and organizational systems involve multiple interacting entities whose relationships evolve over time. These may include:

- Firms in a market with supply-demand relationships.
- Departments in an organization exchanging resources.
- Countries engaged in trade or political alliances.

MAFENON can simulate:

- Dynamic market interactions under changing external conditions.
- Resource allocation processes with temporal constraints.
- Strategic decision-making with feedback loops between agents.

This modeling capacity supports applications in business optimization, policy planning, and resilience analysis for complex economic systems.

Overall, these examples underscore MASFENON's potential as a general-purpose, high-performance simulation platform for temporally evolving, multi-agent, network-based systems across multiple domains.

Conclusions and Outlook

This doctoral research has been a formative experience in advancing my expertise across several interconnected domains: **mathematical modeling, simulation of complex systems, chaos theory**, and, most significantly, **distributed programming and high-performance computing (HPC)**. The development of the MASFENON framework brought together these strands into a coherent, extensible platform capable of simulating diverse, dynamic, and interconnected networks with high computational efficiency.

From a scientific perspective, MASFENON contributes a flexible and modular approach to multi-agent simulation, bridging theoretical modeling with practical implementation on large-scale HPC systems. Its architecture supports a wide spectrum of use cases, from the simulation of cell-to-cell communication networks and epidemic spread to sociological and economic dynamics. This breadth reflects not only the generality of the underlying mathematical formulations but also the deliberate design choices made to enable scalability, adaptability, and integration with heterogeneous datasets.

On the computational side, the work carried out has deepened my understanding of **parallel and distributed algorithm design, MPI-based inter-process communication**, and **shared-memory parallelism** using **OpenMP**, alongside experimentation with GPU acceleration and heterogeneous architectures. The iterative process of optimization, profiling, and performance tuning has shaped my perspective on implementing real-world simulation frameworks that can operate efficiently at scale. I also got a lot of experience on complex systems modeling and analysis, so that is also something that was very important during this Ph.D.

Future Directions

The work presented here is not an endpoint but a foundation. In the coming years, I intend to continue pursuing the HPC implementation of large scale, real world simulations, both as a direct continuation of this research and in broader contexts where similar computational and modeling challenges arise.

MAFENON itself will remain an active and evolving project. Many of

the features and extensions envisioned during this research are already documented in the public repository as **GitHub Issues** at <https://github.com/josura/MASFENON/issues>, awaiting further development. Planned enhancements include:

- Full implementation of the missing features outlined in the design phase, such as advanced intra-temporal dynamics, hypergraph support, and finer-grained state representations.
- Integration of more sophisticated parameter fitting pipelines, including multi-parameter and agent-specific scaling optimization.
- Expanded visualization and analysis interfaces for interactive exploration of simulation outputs.
- Further HPC-oriented optimizations to prepare the framework for **exascale computing environments**.

Contributions and Acknowledgments

This research would not have been possible without the contributions, guidance, and support of numerous individuals and institutions:

- **Prof. Alfredo Pulvirenti**, my supervisor, for his invaluable guidance in shaping this research, supporting publications, facilitating access to HPC resources, and providing continuous feedback throughout the Ph.D.
- **Prof. Andrea Rapisarda**, my Ph.D. coordinator, for his constant availability in addressing questions related to the doctoral program, missions, and funding opportunities.
- My collaborators in the Catania bioinformatics group: **Antonio Di Maria**, **Grete Privitera**, **Prof. Salvatore Alaimo**, and **Prof. Alfredo Ferro**, for their insights, discussions, and technical contributions.
- My Ph.D. colleagues: **Viviana Condorelli**, with whom I conducted research in sociology, and **Ilaria Cosentini**, with whom I explored Boolean Networks.
- The **NextGenerationEU** funding program, for financially supporting my doctoral research.
- **CINECA** for giving me computational resources on the **Leonardo supercomputer**.
- My colleagues at **Teesside University, UK**: **Prof. Claudio Angione** and **Prof. Annalisa Occhipinti**, for hosting me during my international research period, guiding me through the work there, and offering valuable opportunities for collaboration. I also thank **Doan Le Minh Thao**, **Suraj Verma**, and **Kaveh Shahhosseini** for their assistance with metabolic data, as well as for the many stimulating discussions and brainstorming sessions. Special thanks to **Farhan Ah-**

mad and **Mario Verdicchio** for their companionship and for joining me in a hackathon that significantly enriched my HPC experience.

- Finally, I wish to acknowledge the assistance of **large language models (LLMs)**, including the one supporting the writing of some parts of this thesis, for helping me articulate my work in clear and comprehensible and clearer English, which is not my primary language, and for improving the clarity and precision of my communication.

Writing Process and Use of Large Language Models

The writing process of this thesis followed a structured and iterative methodology. First, a complete skeleton structure of the document was defined, outlining the logical flow between chapters and sections. Each part of the thesis was then developed, starting from a draft entirely written by the author, based either on new documentation or on previous materials such as internal reports and published articles. These drafts contained the conceptual logic, methodological explanations, and the scientific narrative intended for each section.

Once a draft was completed, Large Language Models (LLMs), such as ChatGPT or locally deployed alternatives, were employed as writing assistants. They were used to improve clarity, conciseness, and stylistic consistency through targeted prompts (e.g., "make this section more understandable/concise/consistent", or "address these issues in this part"). In some cases, LLMs were also prompted to suggest ways to integrate additional related concepts or improve the logical connection between sections.

The generated suggestions were always **critically reviewed**, **refined**, and **edited** to ensure logical consistency, technical accuracy, and coherence with the scientific and methodological framework of the thesis. No content was included without full verification and adaptation to the context of the work. This iterative approach allowed for a clearer and more readable final text, while maintaining full intellectual authorship and scientific rigor.

Final Remarks

Beyond its specific applications, MASFENON demonstrates that it is possible to design a simulation framework that is both theoretically rigorous and computationally efficient, while remaining sufficiently modular to adapt to entirely new domains of inquiry. My long-term goal is to extend this approach, applying HPC-driven modeling to address pressing scientific and societal challenges, from understanding biological complexity to anticipating the dynamics of large-scale social and technological systems.

The journey undertaken during this Ph.D. has reinforced the importance of combining strong theoretical foundations with advanced computational methodologies. It has also confirmed my commitment to advancing the

frontiers of HPC-enabled simulation science, both through continued development of MASFENON and through new research endeavors that build on the skills, insights, and tools developed here.

On a personal level, this work has deepened my passion for research in mathematics, physics, computer science, and the study of complex systems. It has also expanded my expertise in chaos theory, optimization, and distributed programming. I am profoundly grateful for the opportunities made possible through European Union funding and the supportive research environment in Europe, which enabled me to pursue this project. Looking ahead, I am committed to continuing my career as a European researcher, contributing to future projects and proposals at the intersection of HPC and complex systems science.

Appendix A

Appendix: Mathematical Foundation

This appendix section is included for completeness, providing readers, particularly those with a theoretical or computational background, with a formal overview of the linear algebra concepts, graph theory and the mathematical background of MASFENON most relevant features to this work.

A.1 Linear algebra

Linear algebra constitutes a fundamental pillar of the computational and analytical backbone of the MASFENON framework. While the main body of this thesis focuses on the biological, algorithmic, and high-performance computing aspects of the system, the matrix-based formulations that underpin many of the simulation steps deserve explicit mention and clarification.

Although it may not be essential for all readers to engage with these details, a solid understanding of the linear algebraic structures used (e.g., adjacency matrices, Laplacians, pseudoinverses, eigen decompositions, and sparse matrix formats) helps to appreciate the design choices made in the implementation of MASFENON, especially those involving state updates, network transformations, graph metrics, and inter-agent communication modeling.

We will focus on both foundational definitions and their practical interpretations in a simulation setting, often linking back to how these abstractions are concretely applied in modeling and analyzing large-scale dynamic systems within the MASFENON pipeline.

A.1.1 Vectors and Vector Spaces

Vectors are fundamental elements of linear algebra, commonly used to represent quantities that possess both magnitude and direction. In the context of simulation frameworks such as MASFENON, vectors are employed to model

a wide range of entities, including agent states, node attributes in networks, and time-dependent signals evolving over simulation steps.

Throughout this appendix, we adopt two common notational conventions to denote vectors: the boldface style (e.g., $\mathbf{v}$) and the arrow notation (e.g., $\vec{v}$). These notations are used interchangeably, as both are widely recognized and mathematically equivalent. However, the choice of notation can significantly impact readability, especially when dealing with complex expressions involving multiple vectorial terms or nested operations.

To enhance visual clarity and reduce ambiguity in such cases (such as when explaining the mathematical background of MASFENON or some graph theory concepts), we prefer the use of the arrow notation $\vec{v}$, which offers a more explicit indication of vectorial nature, especially in handwritten derivations or multi-line equations. This notational flexibility serves not only to improve comprehension but also to maintain consistency with graphical or physical interpretations commonly used in modeling agent-based systems and dynamical processes in networks.

Definition (Vector). A vector over a field $\mathbb{F}$ (typically $\mathbb{R}$ or $\mathbb{C}$) is an ordered tuple of elements from that field. Formally, an n -dimensional vector is an element of the vector space $\mathbb{F}^n$ and is written as:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}, \quad \text{with } v_i \in \mathbb{F}$$

Definition (Vector Space). A vector space V over a field $\mathbb{F}$ is a set equipped with two operations:

- *Vector addition:* $+: V \times V \rightarrow V$
- *Scalar multiplication:* $\cdot: \mathbb{F} \times V \rightarrow V$

such that the following axioms hold for all $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ and all scalars $a, b \in \mathbb{F}$:

1. $\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$ (associativity)
2. $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$ (commutativity)
3. There exists a zero vector $\mathbf{0} \in V$ such that $\mathbf{v} + \mathbf{0} = \mathbf{v}$
4. For every $\mathbf{v} \in V$, there exists an inverse $-\mathbf{v} \in V$ such that $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$
5. $a \cdot (\mathbf{u} + \mathbf{v}) = a \cdot \mathbf{u} + a \cdot \mathbf{v}$
6. $(a + b) \cdot \mathbf{v} = a \cdot \mathbf{v} + b \cdot \mathbf{v}$
7. $a \cdot (b \cdot \mathbf{v}) = (ab) \cdot \mathbf{v}$
8. $1 \cdot \mathbf{v} = \mathbf{v}$ (where 1 is the multiplicative identity in $\mathbb{F}$)

Examples.

- The set $\mathbb{R}^n$ is a vector space under elementwise addition and scalar multiplication.

- The set of all n -dimensional vectors representing gene expression profiles across different samples.
- In MASFENON, the states of N agents over time can be represented as a vector $\mathbf{x}(t) \in \mathbb{R}^N$, where each entry $x_i(t)$ denotes the internal state or concentration value of the i -th agent at time t .

Remarks. Vector spaces form the foundation upon which all other linear algebraic structures are built. The notions of basis, dimension, subspaces, and orthogonality, introduced in the following sections, rely on this structure. In the context of high-performance simulations, vectors are often stored as sparse structures or streams, depending on memory and computational constraints.

A.1.2 Linear Independence, Basis, and Dimension

Understanding how vectors relate to each other is essential in both theoretical and computational linear algebra. In particular, the concepts of linear independence, basis, and dimension allow one to describe the structure of a vector space concisely and efficiently.

Definition (Linear Independence). A set of vectors $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k\}$ in a vector space V is said to be **linearly independent** if the only solution to the equation

$$a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \dots + a_k\mathbf{v}_k = \mathbf{0}$$

is $a_1 = a_2 = \dots = a_k = 0$.

If a non-trivial combination (some $a_i \neq 0$) exists that results in the zero vector, then the set is said to be **linearly dependent**.

Definition (Span). The **span** of a set of vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is the set of all linear combinations of those vectors:

$$\text{span}(\mathbf{v}_1, \dots, \mathbf{v}_k) = \left\{ \sum_{i=1}^k a_i \mathbf{v}_i \mid a_i \in \mathbb{F} \right\}$$

This span forms a subspace of V .

Definition (Basis). A basis of a vector space V is a set of vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ that:

1. Are linearly independent.
2. Span the entire space V .

Every element $\mathbf{v} \in V$ can be written uniquely as a linear combination of basis vectors. A basis provides a coordinate system for the space.

Definition (Dimension). The **dimension** of a vector space V is the number of vectors in any basis of V . If V has a finite basis with n vectors, then V is said to be n -dimensional, written as $\dim(V) = n$.

Example. Let $V = \mathbb{R}^3$. The standard basis is:

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

These vectors are linearly independent and span $\mathbb{R}^3$, so they form a basis, and $\dim(\mathbb{R}^3) = 3$.

Context in MASFENON. In simulations of multi-agent systems or biological networks, the dimensionality of the state space often grows rapidly with the number of nodes, layers, or agents. Efficient computation of a reduced basis (e.g., via PCA or SVD, as will be introduced later) is frequently necessary to project high-dimensional data into lower-dimensional spaces while preserving essential dynamics. The basis vectors in this reduced space often represent principal modes or archetypes of system behavior.

A.1.3 Change of Basis

Once a basis for a vector space has been chosen, every vector can be uniquely expressed in terms of its coordinates relative to that basis. However, in both theory and applications, particularly in simulations, dimensionality reduction, or model transformations, it is often beneficial to change the basis used to represent vectors or transformations.

Definition (Change of Basis). Let $\mathcal{B} = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ and $\mathcal{B}' = \{\mathbf{w}_1, \dots, \mathbf{w}_n\}$ be two different bases for the same n -dimensional vector space V . The **change of basis matrix** $\mathbf{P}$ transforms coordinates from $\mathcal{B}'$ to $\mathcal{B}$:

$$[\mathbf{x}]_{\mathcal{B}} = \mathbf{P}[\mathbf{x}]_{\mathcal{B}'}.$$

Equivalently,

$$[\mathbf{x}]_{\mathcal{B}'} = \mathbf{P}^{-1}[\mathbf{x}]_{\mathcal{B}}.$$

Here, $[\mathbf{x}]_{\mathcal{B}}$ and $[\mathbf{x}]_{\mathcal{B}'}$ are coordinate representations of the same vector $\mathbf{x}$ in bases $\mathcal{B}$ and $\mathcal{B}'$, respectively.

Matrix Representation. The columns of $\mathbf{P}$ are the vectors of the new basis $\mathcal{B}'$ expressed in terms of the old basis $\mathcal{B}$:

$$\mathbf{P} = \begin{bmatrix} | & | & \dots & | \\ \mathbf{w}_1 & \mathbf{w}_2 & \dots & \mathbf{w}_n \\ | & | & & | \end{bmatrix}_{\mathcal{B}}.$$

Matrix Similarity. Changing basis also affects how linear transformations are represented. If $\mathbf{A}$ is a matrix representing a linear transformation in basis $\mathcal{B}$, then its representation $\mathbf{A}'$ in basis $\mathcal{B}'$ is given by:

$$\mathbf{A}' = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}.$$

This operation, called a similarity transformation, preserves key properties of $\mathbf{A}$, such as its eigenvalues and determinant, while changing the basis in which it acts.

Geometric Interpretation. Changing basis is akin to rotating or skewing your "coordinate grid." For example, a vector $\mathbf{x}$ might be expressed as $(1, 2)$ in the standard basis, but its coordinates could be $(\sqrt{2}, 0)$ in a rotated basis aligned with the diagonal of the unit square. While the vector in space remains the same, its numerical description changes depending on the "axes" we choose.

Numerical Example. Suppose we work in $\mathbb{R}^2$ with the standard basis:

$$\mathcal{B} = \left\{ \mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}, \quad \mathcal{B}' = \left\{ \mathbf{w}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{w}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\}.$$

To express a vector $\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ in the new basis $\mathcal{B}'$, we write:

$$\mathbf{x} = a\mathbf{w}_1 + b\mathbf{w}_2 = a \begin{bmatrix} 1 \\ 1 \end{bmatrix} + b \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} a - b \\ a + b \end{bmatrix}.$$

Solving for a, b from

$$\begin{bmatrix} a - b \\ a + b \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix},$$

we find $a = \frac{3}{2}, b = \frac{-1}{2}$, so:

$$[\mathbf{x}]_{\mathcal{B}'} = \begin{bmatrix} \frac{3}{2} \\ -\frac{1}{2} \end{bmatrix}.$$

Context in MASFENON. In MASFENON, change of basis is particularly relevant when transitioning between different representations of agent states or dynamic modes. For example:

- In spectral graph analysis, we move to the eigenbasis of the Laplacian matrix to study smoothness or diffusion.
- In model reduction, reduction of dimensionality and visualization of the data that we use, high-dimensional vectors are projected onto a lower-dimensional principal subspace using PCA or SVD.
- When integrating experimental data, gene expression patterns may be interpreted differently in a latent factor basis (e.g., transcription factor activity space).

Further Reading. Change of basis is deeply connected to diagonalization, matrix similarity, and invariant subspaces. These are discussed in Appendix A.1.4 (Eigenvalues and Eigenvectors) and Appendix A.4 (Matrix Decompositions).

A.1.4 Inner Products, Norms, and Orthogonality

Inner products, norms, and orthogonality form the foundation for many theoretical and computational tools in linear algebra. They provide a way to measure angles, distances, and similarity between vectors, playing a critical role in algorithms used throughout data science, computational biology, and multi-agent simulation frameworks like MASFENON.

Definition (Inner Product). An **inner product** on a real vector space V is a function

$$\langle \cdot, \cdot \rangle : V \times V \rightarrow \mathbb{R}$$

that satisfies the following properties for all vectors $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ and scalar $a \in \mathbb{R}$:

1. **Symmetry:** $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle$.
2. **Linearity in the first argument:** $\langle a\mathbf{u} + \mathbf{v}, \mathbf{w} \rangle = a\langle \mathbf{u}, \mathbf{w} \rangle + \langle \mathbf{v}, \mathbf{w} \rangle$.
3. **Positive definiteness:** $\langle \mathbf{v}, \mathbf{v} \rangle \geq 0$, and $\langle \mathbf{v}, \mathbf{v} \rangle = 0 \iff \mathbf{v} = \mathbf{0}$.

The inner product induces a natural way to measure angles and distances between vectors.

Definition (Norm). The norm of a vector $\mathbf{v}$ in an inner product space is defined as:

$$\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$$

This definition generalizes the concept of length or magnitude, and it satisfies the following properties:

1. **Non-negativity:** $\|\mathbf{v}\| \geq 0$, and $\|\mathbf{v}\| = 0 \iff \mathbf{v} = \mathbf{0}$.
2. **Homogeneity:** $\|a\mathbf{v}\| = |a|\|\mathbf{v}\|$ for scalar a .
3. **Triangle inequality:** $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$.

Definition (Orthogonality). Two vectors $\mathbf{u}, \mathbf{v}$ are **orthogonal** if their inner product is zero:

$$\langle \mathbf{u}, \mathbf{v} \rangle = 0$$

A set of vectors is called **orthonormal** if each vector has unit norm and all pairs of distinct vectors in the set are orthogonal.

Example (Dot Product in $\mathbb{R}^n$). In the Euclidean space $\mathbb{R}^n$, the most common inner product is the dot product, defined as:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \mathbf{y} = \sum_{i=1}^n x_i y_i$$

Application in Computational Methods. Orthogonality and norms underpin many powerful numerical methods such as Singular Value Decomposition (SVD), Principal Component Analysis (PCA), and Orthogonal Projections, techniques extensively used within MASFENON. Specifically, inner product spaces allow algorithms to quantify similarity and measure deviations in high-dimensional spaces, which are crucial for dimension reduction, signal processing, and network simulations.

Context in MASFENON. In MASFENON, orthogonality and inner products are very important for invertible matrices, and we will need to work with matrices that should be invertible. can assist in isolating independent patterns of behavior or interaction among agents. For example, orthogonal projections can be employed to separate different functional pathways or independent signal streams in complex biological networks or multi-agent systems.

A.1.5 Matrices and Operations

Matrices provide an essential framework for representing linear transformations and systems of linear equations. This subsection introduces key definitions, matrix operations, and foundational concepts crucial for understanding and efficiently utilizing linear algebra within the MASFENON simulation framework.

Matrix Definitions. A **matrix** is a rectangular array of elements arranged in rows and columns. A matrix $\mathbf{A}$ with m rows and n columns is written as an $m \times n$ matrix:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}, \quad a_{ij} \in \mathbb{R}.$$

Common matrix types include:

- **Square Matrix:** $m = n$, same number of rows and columns.
- **Diagonal Matrix:** Square matrix where all off-diagonal elements are zero, i.e. $a_{ij} = 0$ if $i \neq j$.

- **Identity Matrix:** Special diagonal matrix with ones on the diagonal, denoted $\mathbf{I}_n$ for an $n \times n$ matrix.
- **Symmetric Matrix:** Square matrix satisfying $\mathbf{A}^\top = \mathbf{A}$.
- **Sparse Matrix:** Matrix where most elements are zero.

Core Matrix Operations. Given matrices $\mathbf{A} = (a_{ij})$, $\mathbf{B} = (b_{ij})$ and scalar c , core operations include:

1. **Addition:** For two matrices of the same dimensions,

$$(\mathbf{A} + \mathbf{B})_{ij} = a_{ij} + b_{ij}$$

2. **Scalar Multiplication:** Multiplying each element by scalar c ,

$$(c\mathbf{A})_{ij} = ca_{ij}$$

3. **Matrix Multiplication:** Given $\mathbf{A}$ of size $m \times p$ and $\mathbf{B}$ of size $p \times n$, their product $\mathbf{C} = \mathbf{AB}$ has dimensions $m \times n$ and is given by:

$$c_{ij} = \sum_{k=1}^p a_{ik}b_{kj}$$

4. **Transpose:** For a matrix $\mathbf{A}$, its transpose $\mathbf{A}^\top$ is defined by interchanging rows and columns,

$$(\mathbf{A}^\top)_{ij} = a_{ji}$$

5. **Inverse:** A square matrix $\mathbf{A}$ is invertible (or non-singular) if there exists a matrix $\mathbf{A}^{-1}$ such that:

$$\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

Rank and Null Space. The **rank** of a matrix $\mathbf{A}$, denoted as $\text{rank}(\mathbf{A})$, is the number of linearly independent rows or columns. The **null space** (or kernel) of $\mathbf{A}$, denoted as $\text{Null}(\mathbf{A})$, consists of all solutions $\mathbf{x}$ to the equation:

$$\mathbf{Ax} = \mathbf{0}$$

The dimension of the null space is given by:

$$\dim(\text{Null}(\mathbf{A})) = n - \text{rank}(\mathbf{A})$$

Solving Linear Systems (Gaussian Elimination). A system of linear equations can be represented compactly as a matrix equation:

$$\mathbf{Ax} = \mathbf{b},$$

where $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{x}$ an $n \times 1$ vector of unknowns, and $\mathbf{b}$ an $m \times 1$ vector.

A fundamental method for solving linear systems is **Gaussian Elimination**, which transforms the augmented matrix $[\mathbf{A}|\mathbf{b}]$ to a row-echelon form through row operations:

1. Interchange two rows,
2. Multiply a row by a nonzero scalar,
3. Add a scalar multiple of one row to another row.

After forward elimination, back substitution can be applied to obtain solutions explicitly. Gaussian elimination can also be extended to compute matrix inverses and determinants efficiently.

Context in MASFENON. Matrix operations and solving linear systems are central computational routines in MASFENON. Specifically, large-scale simulations often involve sparse matrix operations, rank computation, and inverse operations to solve linear constraints and optimization problems efficiently. Computational performance is significantly enhanced through advanced numerical methods leveraging sparsity, parallelization, and optimized libraries.

Reference. Detailed theoretical explanations, proofs, and numerical algorithms for Gaussian elimination, rank computation, and matrix inversion, including the LU and QR decompositions, can be found in Appendix A.1.7.

A.1.6 Eigenvalues and Eigenvectors

Eigenvalues and eigenvectors are fundamental concepts in linear algebra, especially relevant for analyzing matrix behavior and stability properties of dynamical systems. They provide essential insights into the structure of linear transformations and play a critical role in various numerical algorithms and simulations, particularly in the MASFENON framework.

Definitions. Consider a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$. A non-zero vector $\mathbf{x} \in \mathbb{R}^n$ is called an **eigenvector** of $\mathbf{A}$ if there exists a scalar $\lambda \in \mathbb{R}$ such that:

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}.$$

Here, λ is termed the **eigenvalue** associated with the eigenvector $\mathbf{x}$.

Characteristic Equation. To find eigenvalues and eigenvectors, we solve the characteristic equation derived from:

$$\mathbf{A}\mathbf{x} - \lambda\mathbf{x} = (\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}.$$

For a nontrivial solution $\mathbf{x}$ (i.e., $\mathbf{x} \neq \mathbf{0}$), the matrix $(\mathbf{A} - \lambda\mathbf{I})$ must be singular. Thus, we obtain the characteristic polynomial:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0.$$

Solving this polynomial equation yields the eigenvalues λ_i of the matrix $\mathbf{A}$.

Diagonalization. A matrix $\mathbf{A}$ is said to be diagonalizable if there exists an invertible matrix $\mathbf{P}$ and a diagonal matrix $\mathbf{D}$ such that:

$$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1},$$

where $\mathbf{D}$ contains eigenvalues of $\mathbf{A}$ on the diagonal, and the columns of $\mathbf{P}$ are the corresponding eigenvectors.

Diagonalization simplifies numerous matrix operations, such as raising matrices to powers:

$$\mathbf{A}^k = \mathbf{P}\mathbf{D}^k\mathbf{P}^{-1}.$$

Spectral Decomposition. For symmetric matrices ($\mathbf{A} = \mathbf{A}^\top$), the spectral decomposition theorem states that $\mathbf{A}$ is orthogonally diagonalizable:

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top,$$

where $\mathbf{Q}$ is an orthogonal matrix whose columns are orthonormal eigenvectors of $\mathbf{A}$, and $\mathbf{\Lambda}$ is a diagonal matrix containing eigenvalues. This decomposition is critical in principal component analysis (PCA) and other dimensionality reduction methods frequently used in bioinformatics and systems biology.

Applications in Stability Analysis. Eigenvalues are fundamental for the stability analysis of linear dynamical systems. Given a discrete-time linear system defined by:

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t),$$

the behavior of the system is determined by the eigenvalues of $\mathbf{A}$:

- If $|\lambda_i| < 1$ for all i , the system converges to a stable equilibrium.
- If any eigenvalue satisfies $|\lambda_i| > 1$, the system diverges, indicating instability.

Similarly, for continuous-time linear systems given by:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t),$$

the stability is determined by the real parts of the eigenvalues:

- If $\text{Re}(\lambda_i) < 0$ for all i , the system is stable.
- If any eigenvalue satisfies $\text{Re}(\lambda_i) > 0$, the system is unstable.

Context in MASFENON. In the MASFENON framework, eigenvalues and eigenvectors play a central role in both the mathematical foundations and the computational strategies of the simulation engine. They are employed to analyze stability of large-scale interconnected systems, to characterize propagation dynamics, and to optimize the handling of high-dimensional state spaces. Applications include eigenvector centrality for identifying influential nodes, spectral clustering for community detection, and principal component analysis for dimensionality reduction.

Most importantly, eigen-decomposition underlies the computation of the **Moore-Penrose pseudoinverse**, which is a cornerstone operation in MASFENON. The pseudoinverse is required to propagate signals across heterogeneous networks when exact inverses do not exist or are numerically unstable. Since the pseudoinverse is defined via singular value decomposition (SVD), which itself relies on eigenvalues and eigenvectors of associated matrices, understanding these concepts is crucial for grasping how MASFENON implements propagation, dissipation, and conservation dynamics at scale.

Thus, while eigenvalues and eigenvectors are classical tools in linear algebra, in the context of MASFENON they are directly connected to the core mechanism of simulation: enabling the computation of generalized inverses on complex, often singular, network topologies.

References. Further mathematical details and advanced numerical methods related to eigenvalue computation, including Moore-Penrose inverse and QR algorithms, can be found in Appendix A.1.7.

A.1.7 Matrix Decompositions and Inverse

Matrix decompositions play a key role in simplifying complex matrix operations and understanding matrix structures.

Eigen-decomposition for Symmetric Matrices. For symmetric matrices ($\mathbf{A} = \mathbf{A}^\top$), the eigen-decomposition simplifies into a spectral decomposition:

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top,$$

where $\mathbf{Q}$ is an orthogonal matrix composed of eigenvectors, and $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues. This decomposition is fundamental in Principal Component Analysis (PCA) and network analyses.

Singular Value Decomposition (SVD). Any real matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ can be factorized via the Singular Value Decomposition (SVD) into:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top,$$

where $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is a diagonal matrix containing singular values σ_i . SVD is extensively used in dimensionality reduction, data compression, and noise reduction.

QR Decomposition. A matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ can be decomposed into a product of an orthogonal matrix $\mathbf{Q}$ and an upper-triangular matrix $\mathbf{R}$:

$$\mathbf{A} = \mathbf{Q}\mathbf{R}.$$

Common numerical methods to obtain QR decomposition include Householder reflections and Givens rotations, both important for numerical stability and efficiency in solving linear systems and eigenvalue problems.

Inverse and Moore-Penrose Pseudoinverse: Matrix Inverse. A square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is invertible if $\det(\mathbf{A}) \neq 0$. Its inverse $\mathbf{A}^{-1}$ satisfies:

$$\mathbf{A} \mathbf{A}^{-1} = \mathbf{A}^{-1} \mathbf{A} = \mathbf{I}.$$

Common computational methods include:

- *Gaussian elimination* via augmented matrices,
- *Adjugate/determinant formula* (practical only for small matrices),
- *Eigenvalue decomposition*, if $\mathbf{A}$ is diagonalizable: $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$, then $\mathbf{A}^{-1} = \mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Q}^{-1}$.

Moore-Penrose Pseudoinverse. For singular or rectangular matrices, $\mathbf{A}^+ \in \mathbb{R}^{n \times m}$ generalizes the inverse. It is uniquely defined by the four Penrose conditions:

$$\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A}, \quad \mathbf{A}^+\mathbf{A}\mathbf{A}^+ = \mathbf{A}^+, \quad (\mathbf{A}\mathbf{A}^+)^T = \mathbf{A}\mathbf{A}^+, \quad (\mathbf{A}^+\mathbf{A})^T = \mathbf{A}^+\mathbf{A}$$

A numerically stable way to compute it is via the singular value decomposition (SVD): if $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, then

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

where $\mathbf{\Sigma}^+$ is obtained by reciprocating non-zero singular values and transposing the result.

In special cases:

$$\mathbf{A}^+ = \begin{cases} (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T, & \text{if } \mathbf{A} \text{ has full column rank,} \\ \mathbf{A}^T (\mathbf{A} \mathbf{A}^T)^{-1}, & \text{if } \mathbf{A} \text{ has full row rank.} \end{cases}$$

Use in least squares and minimum-norm solutions. The pseudoinverse gives the least squares solution:

$$\mathbf{x} = \mathbf{A}^+ \mathbf{b}$$

minimizing $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ and, among all solutions, has minimal $\|\mathbf{x}\|_2$.

Numerical stability considerations Computing the pseudoinverse by forming $\mathbf{A}^T \mathbf{A}$ is numerically unstable, as the condition number is squared. The SVD-based method is preferred for stability. The Moore-Penrose computation is inherently expensive, particularly for large matrices, highlighting the importance of choosing computationally appropriate strategies.

Relation to MASFENON As we have seen in the main text in chapter 3, one of the propagation formulation that was declared in 3.3 was the following:

$$(I - \overline{W}^T) \vec{v} = \vec{Input}, \quad \vec{v} = (I - \overline{W}^T)^+ \vec{Input},$$

this aligns directly with the minimum-norm least squares solution interpretation of the pseudoinverse. Gaussian inversions alone do not generalize; the pseudo-inverse elegantly handles both singular and rectangular cases.

A.1.8 Applications to Networks & Graphs

Linear algebra provides essential tools for analyzing and interpreting network structures.

Adjacency and Laplacian Matrices. Networks and graphs can be represented by matrices. The adjacency matrix $\mathbf{A}$ of a graph $G = (V, E)$ with vertices V and edges E is defined as:

$$A_{ij} = \begin{cases} 1 & (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

The Laplacian matrix $\mathbf{L}$ is defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{A},$$

where $\mathbf{D}$ is the diagonal matrix of vertex degrees.

Interpretation of Eigenvalues and Eigenvectors in Graphs. Eigenvalues of the Laplacian matrix provide insights into the connectivity and robustness of the graph. For instance:

- The multiplicity of eigenvalue 0 corresponds to the number of connected components.
- Small eigenvalues reflect weak connections or potential bottlenecks.

Eigenvectors associated with small eigenvalues are used in spectral clustering algorithms.

Sparse Matrix-Vector Products for Network Simulations. Large-scale network simulations heavily rely on sparse matrix representations to efficiently compute matrix-vector products:

$$\mathbf{y} = \mathbf{A}\mathbf{x},$$

where $\mathbf{A}$ is typically sparse. Efficient computation directly impacts performance in large network simulations.

A.1.9 Advanced Topics

Determinants and their Properties. The determinant of a square matrix $\mathbf{A}$, denoted $\det(\mathbf{A})$ or $|\mathbf{A}|$, provides insights about matrix invertibility and volume scaling under linear transformations:

$\mathbf{A}$ is invertible if and only if $\det(\mathbf{A}) \neq 0$.

Important properties include multiplicativity ($\det(\mathbf{AB}) = \det(\mathbf{A}) \det(\mathbf{B})$) and invariance under transposition ($\det(\mathbf{A}) = \det(\mathbf{A}^\top)$).

Matrix Functions. Functions of matrices, notably the matrix exponential $\exp(\mathbf{A})$, defined by the series expansion:

$$\exp(\mathbf{A}) = \sum_{k=0}^{\infty} \frac{\mathbf{A}^k}{k!},$$

are fundamental in solving differential equations and modeling dynamical systems. The trace, $\text{tr}(\mathbf{A})$, is another useful function defined as the sum of diagonal elements and relates closely to the eigenvalues:

$$\text{tr}(\mathbf{A}) = \sum_i \lambda_i.$$

Brief on Tensors and Multilinear Algebra. Tensors are higher-order generalizations of vectors and matrices. Formally, a tensor $\mathcal{T}$ of order n can be viewed as a multidimensional array:

$$\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}.$$

Tensors are essential for representing multidimensional data, and tensor decompositions (e.g., CP and Tucker decompositions) generalize matrix decompositions.

Notation & Conventions

For clarity, the following notational conventions are consistently used throughout this document:

- Matrices: Bold uppercase letters (e.g., $\mathbf{A}, \mathbf{B}$).
- Vectors: Bold lowercase letters (e.g., $\mathbf{x}, \mathbf{y}$) or vector notation (e.g., $\vec{x}, \vec{y}$).
- Scalars: Regular lowercase letters (e.g., a, b, c), Greek letters (α, β, λ).
- Identity matrix: $\mathbf{I}$.
- Transpose of a matrix: $\mathbf{A}^\top$.
- Inner product: $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \mathbf{y}$.
- Norm: $\|\mathbf{x}\|$.

These conventions help maintain readability and consistency throughout the mathematical exposition.

A.2 Graph theory

Graphs are a powerful mathematical abstraction for modeling complex systems composed of interacting entities. In disciplines ranging from biology and sociology to transportation and computer networks, **nodes** (or vertices) represent active components, such as individuals, genes, or routers. While **edges** (or links) denote relationships or interactions between them. This formalism enables us to encapsulate the structure, behavior, and dynamics of a system in a unified framework.

Graph theory provides both intuitive and formal tools to:

- Represent and visualize intricate interactions in a compact form.
- Define exact notions of paths, connectivity, and influence.
- Design efficient algorithms for computation and analysis.
- Support metrics that reveal centrality, clustering, and resilience.

These capabilities make graphs indispensable in the MASFENON framework, where we model communication, diffusion, and structural evolution of agent-based systems.

The remainder of this appendix systematically develops the theoretical foundations needed to articulate and implement advanced graph-based metrics and algorithms. We begin with formal definitions and notation, proceed through structural and topological properties, survey graph variants and types, explore key metrics, and conclude with isomorphism theory and traversal concepts such as spanning trees and path existence.

A.2.1 Graphs as Models for Complex Systems

Many real-world systems, ranging from biological networks and social interactions to transportation and engineered infrastructures, are composed of numerous interacting components whose individual behaviors aggregate into emergent phenomena. Graphs offer a **flexible yet rigorous mathematical abstraction** to represent such systems, where:

$$G = (V, E)$$

consists of a vertex set V , representing the system's entities, and an edge set E , encoding their interactions or relationships.

This abstraction provides the following advantages:

- **Compact Representation:** Regardless of system size, graphs efficiently capture structure and interactions in terms of vertices and edges.
- **Unified Analysis:** Graph theory equips us with well-defined, domain-agnostic notions, such as connectivity, paths, and cycles, that can be applied universally.

- **Algorithmic Efficiency:** Classical algorithms (e.g., BFS, Dijkstra, PageRank) can be directly applied to analyze system properties and dynamics.
- **Insight through Metrics:** Structural metrics like degree, centrality, and clustering unveil roles, hierarchies, and resilience within complex networks.

In the MASFENON framework, graphs are not just static blueprints. They encode both the static architecture of agent interactions and dynamic behaviors such as communication, diffusion, and structural evolution over time. Graph-based metrics are used to compute influence, detect changes, understand connectivity, and monitor systemic stability.

These reasons collectively, representational clarity, mathematical precision, and algorithmic versatility, establish graph theory as the theoretical foundation of the MASFENON toolkit.

A.2.2 Fundamental Definitions and Notation

Let $G = (V, E)$ be a graph, where:

$$V = \{v_1, v_2, \dots, v_n\}$$

is a finite set of *vertices* (or *nodes*), and E is a collection of *edges*, each of which connects two vertices. In an *undirected* graph, edges are unordered pairs $\{u, v\} \in E$; in a *directed* graph (digraph), edges are ordered pairs $(u, v) \in E$, indicating direction.

An edge e is said to be *incident* to its endpoints: if $e = (u, v)$, then e is incident on both u and v . Vertices u and v are *adjacent* (neighbors) if there is an edge between them.

Adjacency Matrix The *adjacency matrix* $A = [a_{ij}]$ of a graph G with $|V| = n$ is the $n \times n$ matrix defined by

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

For *weighted graphs*, the entries a_{ij} may carry weights instead of binary values, or the weight can be associated with a function $w : E \rightarrow \mathbb{R}$. In undirected graphs, A is symmetric ($a_{ij} = a_{ji}$).

Incidence Matrix The *incidence matrix* $B = [b_{ie}]$ relates vertices and edges. If $e_k = (u, v)$, then in an undirected graph:

$$b_{ik} = \begin{cases} 1 & \text{if } v_i \text{ is an endpoint of } e_k, \\ 0 & \text{otherwise,} \end{cases}$$

while in a directed graph, one may use oriented incidence entries $+1$ or -1 to indicate outgoing vs incoming edges.

Neighborhood The (open) *neighborhood* of a vertex v , denoted $N(v)$, is the set of vertices adjacent to v :

$$N(v) = \{u \in V : (u, v) \in E\}.$$

The *closed neighborhood* $N[v]$ includes v itself. These neighborhoods form the foundation for measures like clustering coefficient and local density.

A.2.3 Substructures in Graphs

In graph theory, we often analyze specific substructures of a graph $G = (V, E)$, including various types of subgraphs, as well as walks, paths, and cycles.

Subgraphs A *subgraph* $H = (V_H, E_H)$ of G satisfies $V_H \subseteq V$ and $E_H \subseteq E$, with each edge in E_H connecting only vertices in V_H .

- If $V_H = V$ but $E_H \subset E$, H is a *spanning subgraph* of G .
- If H is obtained by deleting vertices (and the incident edges), it is an *induced subgraph*, denoted $G[V_H]$, containing all original edges between vertices in V_H .
- A *proper subgraph* is any subgraph H such that either $V_H \subset V$ or $E_H \subset E$, but not both equal.

Walks, Trails, Paths, and Cycles A *walk* of length ℓ in G is a sequence

$$v_0, e_1, v_1, e_2, \dots, v_{\ell-1}, e_\ell, v_\ell$$

where each edge e_k connects v_{k-1} and v_k . Vertices and edges may repeat. A walk is *closed* if $v_0 = v_\ell$, and *open* otherwise.

A *trail* is a walk where all edges e_k are distinct. A *path* is a trail in which all vertices are distinct (and consequently edges).

A *cycle* (or simple circuit) is a closed trail in which the only repeated vertex is the start/end vertex; equivalently, a closed path.

Relations between Walks and Paths The fundamental property that any walk between two vertices contains a path is foundational:

$$\exists \text{walk from } u \text{ to } v \Rightarrow \exists \text{path from } u \text{ to } v$$

since repeated vertices in a shortest walk can be bypassed.

Notation and Definitions

walk of length $\ell : v_0 \rightsquigarrow v_\ell$ via ℓ edges,

path : walk with distinct vertices,

cycle : closed path $v_0 = v_\ell$, no other repeats.

A.2.4 Connectivity and Distance Measures

Let $G = (V, E)$ be a graph, assumed connected unless stated otherwise.

Connectivity Two vertices $u, v \in V$ are *connected* if there exists a path in G from u to v . A graph is *connected* if every pair of vertices is connected, otherwise, it is *disconnected*. In directed graphs, we distinguish:

strong connectivity: paths exist both $u \rightarrow v$ and $v \rightarrow u$,

weak connectivity: connected when edges are treated as undirected.

Distance The *distance* $d(u, v)$ between two vertices $u, v \in V$ is the length (number of edges) of a shortest path between them. If no path exists, $d(u, v) = \infty$.

Eccentricity The *eccentricity* of a vertex v is the maximum distance between v and any other vertex:

$$\varepsilon(v) = \max_{u \in V} d(v, u).$$

By convention, $\varepsilon(v) = \infty$ if v belongs to a disconnected component.

Diameter and Radius The *diameter* of G is the largest eccentricity among all vertices:

$$\text{diam}(G) = \max_{v \in V} \varepsilon(v).$$

Equivalently, it is the longest among all shortest paths in G . The *radius* of G is the smallest eccentricity:

$$\text{rad}(G) = \min_{v \in V} \varepsilon(v).$$

Vertices that achieve the radius are called the *center* of the graph.

Key Relationships In any connected graph:

$$\text{rad}(G) \leq \text{diam}(G) \leq 2 \text{rad}(G).$$

For trees, the stronger bounds

$$2 \text{rad}(G) - 1 \leq \text{diam}(G) \leq 2 \text{rad}(G)$$

hold.

Computation - *All-pairs shortest paths* (e.g., Floyd-Warshall) or repeated Dijkstra/BFS compute $d(u, v)$, enabling eccentricity, diameter, and radius in $O(nm + n^2 \log n)$ time. - In *trees*, a two-phase BFS/DFS from an arbitrary vertex and then from the furthest found vertex computes the diameter in linear time.

These measures, connectivity, distance, eccentricity, diameter, and radius, are essential to assessing **reachability**, **centrality**, and **robustness** in networks. Relevant graph algorithms in MASFENON leverage these for analyzing communication paths, network resilience, and dynamic structural evolution.

A.2.5 Types of Graphs

Graph theory provides a rich taxonomy of graph types to model diverse phenomena:

- **Multigraph:** A graph permitting multiple *parallel edges* between the same pair of vertices, with or without self-loops. Formally, a multiset of edges connects vertices u, v possibly more than once.
- **Labeled Graph:** Each vertex or edge is assigned a label or identifier, often modeled as a function $\ell_V : V \rightarrow \Sigma_V$ and/or $\ell_E : E \rightarrow \Sigma_E$. Widely used in pattern matching and isomorphism detection.
- **Attributed Graph:** Vertices and/or edges carry additional data attributes (e.g., real values or vectors). Mathematically, attribute functions $a_V : V \rightarrow \mathbb{R}^d$ or $a_E : E \rightarrow \mathbb{R}^d$.
- **Weighted Graph:** Edges receive numerical weights $w : E \rightarrow \mathbb{R}$, often representing cost, capacity, or strength of interaction.
- **Temporal (Dynamic) Graph:** Edges (and possibly vertices) appear, disappear, or change over time. A common model is a sequence $\{G_t = (V_t, E_t)\}$ or time-stamped edges (u, v, t) .
- **k -Partite Graph:** The vertex set is partitioned into k disjoint independent subsets $V = V_1 \cup \dots \cup V_k$, with no edges within the same subset. A complete k -partite graph connects all possible edges between different partitions.
- **Hypergraph:** A generalization where edges (hyperedges) join any number of vertices. Formally, $G = (V, E)$ with $E \subseteq \mathcal{P}(V)$. Includes directed variants with hyperedges having "tail" and "head" sets.

Summary Table . A summary table is presented here to remember the different types of graph that are mentioned during this text.

Graph Type	Definition
Multigraph	Allow multiple edges between vertex pairs (and loops)
Labeled	Vertices/edges carry labels from Σ
Attributed	Attributes $a_V : V \rightarrow \mathbb{R}^d$, $a_E : E \rightarrow \mathbb{R}^d$
Weighted	Edge weights $w : E \rightarrow \mathbb{R}$
Temporal	Graph evolves over time G_t or edges with timestamps
k -Partite	Partition $V = \bigcup_{i=1}^k V_i$ with no intra-partition edges
Hypergraph	Edges can connect more than 2 vertices; $E \subseteq \mathcal{P}(V)$

These variants offer flexible modeling across domains, from social networks (temporal, attributed), recommendation systems (bipartite), to higher-order biochemical interactions (hypergraphs). In MASFENON, we primarily use *directed weighted attributed graphs*, but the architecture is extensible to support additional graph types in future expansions.

A.2.6 Graph Metrics

We now introduce fundamental structural metrics on graphs, which are implemented in the MASFENON framework.

Degree For a vertex v in a directed or undirected graph, the *degree* $\deg(v)$ is the number of incident edges. In directed graphs it decomposes into:

$$\deg_{\text{in}}(v), \quad \deg_{\text{out}}(v), \quad \deg(v) = \deg_{\text{in}}(v) + \deg_{\text{out}}(v).$$

Strength Centrality In a weighted graph $G = (V, E, w)$, where $w : E \rightarrow \mathbb{R}^+$ assigns a non-negative weight to each edge, the *strength* of a node $v \in V$ is a generalization of degree centrality that accounts for the cumulative weight of edges incident to the node.

For a directed graph, we distinguish between:

- **Out-strength:**

$$s_{\text{out}}(v) = \sum_{(v,u) \in E} w(v, u)$$

which measures the total weight of edges *leaving* vertex v .

- **In-strength:**

$$s_{\text{in}}(v) = \sum_{(u,v) \in E} w(u, v)$$

which measures the total weight of edges *entering* vertex v .

- **Total strength:**

$$s(v) = s_{\text{in}}(v) + s_{\text{out}}(v)$$

which aggregates both incoming and outgoing edge weights.

For an undirected graph, each edge (u, v) is considered bidirectional, and the strength of a node is defined as:

$$s(v) = \sum_{u \in N(v)} w(v, u)$$

The *average strength centrality* over the entire graph is defined as the mean of node strengths across all nodes. For a directed graph, the variants are:

- **Average out-strength:**

$$\bar{s}_{\text{out}} = \frac{1}{|V|} \sum_{v \in V} s_{\text{out}}(v)$$

- **Average in-strength:**

$$\bar{s}_{\text{in}} = \frac{1}{|V|} \sum_{v \in V} s_{\text{in}}(v)$$

- **Average total strength:**

$$\bar{s} = \frac{1}{|V|} \sum_{v \in V} s(v)$$

For undirected graphs, the same formula applies using $s(v)$ as defined for symmetric edges.

These measures are used in MASFENON to quantify the intensity of interactions or signal diffusion associated with each node in a weighted network. Other metrics that can be derived and used to characterise a network are related to the maximum or minimum strength in the graph.

Clustering Coefficient The clustering coefficient quantifies the tendency of a node's neighbors to form connections among themselves, capturing the presence of local "triangles" in a graph.

Let $G = (V, E, w)$ be a graph with vertex set V , edge set E , and a weight function $w : E \rightarrow \mathbb{R}^+$. Let $N(v)$ denote the neighborhood of node v , and $k_v = |N(v)|$ its degree. Let s_v denote the strength of v , defined as:

$$s_v = \sum_{u \in N(v)} w_{vu}.$$

Unweighted Local Clustering Coefficient In an undirected unweighted graph, the local clustering coefficient of a node v is defined as:

$$C(v) = \frac{2T(v)}{k_v(k_v - 1)},$$

where $T(v)$ is the number of triangles that include node v . This measures the fraction of possible connections among neighbors of v that actually exist.

Weighted Local Clustering Coefficient For a weighted graph, the local clustering coefficient is generalized as follows:

$$C_w(v) = \frac{1}{s_v(k_v - 1)} \sum_{\substack{u, t \in N(v) \\ u \neq t}} \frac{w_{vu} + w_{vt}}{2} \cdot a_{ut},$$

where $a_{ut} = 1$ if edge (u, t) exists and 0 otherwise. This formulation incorporates both topological connectivity and edge weights.

Global Clustering Coefficient The global clustering coefficient of a graph is the average of local clustering coefficients over all nodes:

$$C(G) = \frac{1}{|V|} \sum_{v \in V} C(v),$$

and similarly, for the weighted version:

$$C_w(G) = \frac{1}{|V|} \sum_{v \in V} C_w(v).$$

These measures provide insight into the local cohesiveness and modularity of the graph structure, which are critical in understanding the topological resilience and community structure in complex networks used to represent the agent evolution in MASFENON.

Shortest Paths, Distance, and Measures of Reachability The *distance* $d(u, v)$ between vertices u and v is the length (sum of weights or number of edges) of the shortest path connecting them; $d(u, v) = \infty$ if disconnected. Central to many applications are:

- *Average shortest-path length*:

$$\langle d \rangle = \frac{1}{|V|(|V| - 1)} \sum_{u \neq v} d(u, v).$$

- *Diameter of G* :

$$\text{diam}(G) = \max_{u, v \in V} d(u, v).$$

Centrality Measures Centrality quantifies node importance with respect to paths or connectivity:

- *Degree centrality*: $\deg(v)$. - *Closeness centrality* for connected graphs:

$$C_{\text{clo}}(v) = \frac{|V| - 1}{\sum_{u \in V} d(v, u)}.$$

- *Betweenness centrality*:

$$C_{\text{bet}}(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

where σ_{st} is the number of shortest paths between s and t , and $\sigma_{st}(v)$ is the count that pass through v .

- *Eigenvector centrality*: defined by

$$x_v = \frac{1}{\lambda} \sum_{u \in V} a_{vu} x_u,$$

or vector form $\mathbf{Ax} = \lambda \mathbf{x}$, where A is the adjacency matrix, and λ is the largest eigenvalue.

PageRank A variant of eigenvector centrality suitable for directed graphs, PageRank assigns scores $p(v)$ based on incoming links, with damping factor d , using the recursive definition:

$$p(v) = \frac{1-d}{|V|} + d \sum_{u \in \text{pred}(v)} \frac{p(u)}{\deg_{\text{out}}(u)}.$$

These metrics, degree, strength, clustering, shortest paths, and centralities, constitute the core analytics provided by MASFENON. They enable exploration of roles, influence, cohesion, and reachability within complex networked systems.

A.2.7 Structural Equivalence and Matching

This section covers two fundamental concepts in graph comparison: *graph isomorphism* (exact equivalence of two entire graphs) and *matching* (selecting edge-disjoint relationships within a graph), as well as the notion of *structural equivalence* among vertices.

Graph Isomorphism Two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ are *isomorphic*, denoted $G \simeq H$, if there exists a bijection $f : V_G \rightarrow V_H$ such that

$$\{u, v\} \in E_G \quad \text{if and only if} \quad \{f(u), f(v)\} \in E_H.$$

This bijection preserves adjacency and captures the idea that the two graphs are structurally identical up to relabeling of vertices. An isomorphism from a graph to itself is known as an *automorphism* and reveals graph symmetries.

The *Graph Isomorphism Problem*, determining whether such a bijection exists, remains computationally neither proven to be in P nor NP-complete. However, in practice, efficient algorithms exist for many relevant graph families.

Structural Equivalence of Vertices Two vertices $u, v \in V$ are *structurally equivalent*, denoted $u \sim_s v$, if

$$\forall w \neq u, v: (w, u) \in E \Leftrightarrow (w, v) \in E,$$

and similarly for outgoing edges in directed graphs. In other words, u and v share exactly the same neighbor set. Structural equivalence is a strict equivalence relation, partitioning vertices into equivalence classes.

Graph Matching and Matchings A *matching* $M \subseteq E$ in a graph is a set of edges without shared endpoints. In other words, no two edges in M are adjacent. A *maximal matching* is one that cannot be extended by adding more edges without violating this property; a *maximum matching* is a matching of largest possible size. In bipartite graphs, classical algorithms (e.g., Hopcroft-Karp) find maximum matchings efficiently.

A related concept is *graph matching* in the general sense, finding correspondences between two graphs based on structure or attributes, including both exact matching (isomorphism) and approximate (e.g., subgraph isomorphism, graph edit distance).

These concepts are central in MASFENON’s design and are used during modeling phase and post-simulation analysis, to make account for:

- Detecting redundant or equivalent agents/nodes via automorphisms, to account for behavior redundancy during the simulation;
- Identify similar behavior with similar topological structure;

A.2.8 Trees, Traversability, and Optimization

This section presents key concepts in acyclic structures, traversability, and optimization in graphs that are foundational in algorithmic and structural analysis.

Trees A *tree* is a connected undirected graph with no cycles. Equivalently, it can be defined as a connected graph with n vertices and exactly $n-1$ edges, or as a graph where each pair of vertices is connected by exactly one simple path.

Forests and Rooted Trees A *forest* is a disjoint union of trees (an acyclic graph). A *rooted tree* designates a special vertex (the root), where edges often inherit a parent-child orientation; for example, an arborescence is a directed tree with all edges outward from the root.

Spanning Trees Given a connected graph $G = (V, E)$, a *spanning tree* T is an acyclic subgraph that includes all vertices V and a subset of edges $E_T \subset E$ such that $|E_T| = |V| - 1$ and T is connected. Every connected graph has at least one spanning tree.

Minimum Spanning Tree (MST) In a weighted graph $G = (V, E, w)$, a *minimum spanning tree* T is a spanning tree minimizing the total weight:

$$T = \arg \min_{E_T} \sum_{(u,v) \in E_T} w(u, v).$$

Common greedy algorithms include:

- *Kruskal's algorithm*: Sorts edges by ascending weight and adds each edge if it does not form a cycle, until the tree spans all vertices.
- *Prim's algorithm*: Grows the spanning tree one vertex at a time, at each step adding the minimum-weight edge connecting the current tree to a new vertex.

Traversability *Traversability* concerns paths that visit either all edges or all vertices exactly once:

- *Eulerian path (trail)*: A path that traverses every edge exactly once. A graph is Eulerian (has a closed trail) if and only if it is connected and every vertex has even degree. It has an open trail if exactly two vertices have odd degree.
- *Hamiltonian path (cycle)*: A path that visits every vertex exactly once (cycle if it returns to the start). Determining the existence of Hamiltonian paths/cycles is NP-complete in general.

Optimization Context - **MSTs** are used in network design (e.g., cable layout, broadcast trees) to minimize costs while maintaining connectivity. - **Eulerian trails** are relevant when edge coverage is required exactly once (e.g., routing, inspection). - **Hamiltonian paths** align with optimization problems like the Traveling Salesman Problem (TSP), which is NP-hard.

These concepts are instrumental in the MASFENON framework, and are especially important during the modeling phase:

- *Spanning trees* support network simplification and efficient communication structures.
- *MSTs* guide cost-effective connectivity in weighted agent networks.

A.3 Mathematical Foundation of MASFENON

This section consolidates the mathematical constructs that underpin the MASFENON framework, offering a comprehensive and rigorous formalization of the system’s core propagation mechanisms, graph augmentation strategies, and dynamic simulation algorithms. While the main thesis chapters present high-level conceptual and architectural views of MASFENON, the purpose of this appendix is to present the exact mathematical derivations, iterative update rules, and operator definitions used in both intra- and inter-agent communication.

The formulation extends earlier efforts under the C2C-SEPIA methodology, evolving into a general multi-agent and multi-network simulation paradigm. This section includes detailed definitions of graph structures, propagation matrices, communication functions, and correction techniques (e.g., dissipation, normalization, and conservation). It also presents the pseudoinverse-based formulations required for iterative convergence, and discusses computational trade-offs and complexity. Readers interested in the mathematical rigor behind the simulation dynamics and HPC-oriented implementation will find in this section the complete formal foundation that supports MASFENON.

A.3.1 Graph Model and Notation

The MASFENON framework relies on a layered graph-based abstraction to simulate communication and perturbation dynamics across distributed agents, each operating within a structured subnetwork. This subsection defines the graph-theoretic objects used to formalize both intra-agent and inter-agent structures and operations, building on the general concepts provided in Appendix A.2.

Base Graphs for Agents Each agent $Ag \in V(\text{Agents})$ is associated with a local directed, weighted, attributed graph:

$$G_{Ag} = (V(G_{Ag}), E(G_{Ag}), w_{Ag}),$$

where $V(G_{Ag})$ is the set of internal states (e.g., genes, decision units), $E(G_{Ag}) \subseteq V(G_{Ag}) \times V(G_{Ag})$ represents intra-agent dependencies or influence flows, and $w_{Ag} : E(G_{Ag}) \rightarrow \mathbb{R}$ defines edge weights encoding effect strength and sign.

Virtual Input and Output Nodes To enable inter-agent communication, each base graph is augmented with virtual nodes:

- $V_{in}(Ag) = \{v_{in}^{(1)}, \dots, v_{in}^{(q)}\}$: representing input signals from external agents.

- $V_{out}(Ag) = \{v_{out}^{(1)}, \dots, v_{out}^{(z)}\}$: encoding outputs meant for propagation to other agents.

Augmented Graphs The resulting graph structure for an agent is the **augmented graph**:

$$\mathcal{G}_{Ag} = (V(\mathcal{G}_{Ag}), E(\mathcal{G}_{Ag})),$$

where

$$V(\mathcal{G}_{Ag}) = V(G_{Ag}) \cup V_{in}(Ag) \cup V_{out}(Ag), \quad E(\mathcal{G}_{Ag}) = E(G_{Ag}) \cup E_{in}(Ag) \cup E_{out}(Ag),$$

and:

$$\begin{aligned} E_{in}(Ag) &= \{(v_{in}, v_j) \mid v_{in} \in V_{in}(Ag), v_j \in V(G_{Ag})\}, \\ E_{out}(Ag) &= \{(v_i, v_{out}) \mid v_i \in V(G_{Ag}), v_{out} \in V_{out}(Ag)\}. \end{aligned}$$

Agent Communication Graph At a higher level, MASFENON defines a temporal multigraph of agents:

$$\text{Agents} = (V(\text{Agents}), E(\text{Agents})),$$

where each node is an agent Ag_i , and each edge

$$(v_i, v_j, d) \in E(\text{Agents}), \quad d \in \mathcal{D},$$

represents a directed interaction from agent Ag_i to agent Ag_j occurring at time or duration d , which may be instantaneous or interval-based.

Inter-Agent Communication During simulation, values from the virtual output nodes of an agent Ag_i are transmitted to the corresponding virtual input nodes of another agent Ag_j if $(Ag_i, Ag_j, d) \in E(\text{Agents})$ is active at time t_n . This mechanism implements the coupling of otherwise independent subnetworks into a globally interactive system.

Matrix Representation Each graph $\mathcal{G}_{Ag}$ has an associated **normalized weighted adjacency matrix**:

$$\overline{W}(\mathcal{G}_{Ag}) \in \mathbb{R}^{|V(\mathcal{G}_{Ag})| \times |V(\mathcal{G}_{Ag})|},$$

where:

$$\overline{w}_{ij} = \begin{cases} \frac{w(i, j)}{\sum_{k \in D(i)} |w(i, k)|} & \text{if } w(i, j) \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

This normalization ensures bounded contribution from each node, enabling numerical stability during propagation. This formulation is different from the

one seen in 3.1, but the results are the same. With this formulation, if there is at least one successor edge that is different than 0, then the denominator will be different than 0, as seen in the following proof:

We show that the normalization definition given in Eq. 3.1 and the alternative formulation provided here yield identical results. Consider the two definitions explicitly:

The original formulation (Eq. 3.1):

$$\bar{w}_{ij} = \begin{cases} \frac{w(i, j)}{\sum_{k \in D(i)} |w(i, k)|}, & \text{if } \sum_{k \in D(i)} |w(i, k)| \neq 0, \\ 0, & \text{otherwise.} \end{cases}$$

The alternative formulation used in this appendix:

$$\bar{w}_{ij} = \begin{cases} \frac{w(i, j)}{\sum_{k \in D(i)} |w(i, k)|}, & \text{if } w(i, j) \neq 0, \\ 0, & \text{otherwise.} \end{cases}$$

Proof:

1. If $w(i, j) = 0$, then by both definitions clearly $\bar{w}_{ij} = 0$.
2. If there exists at least one successor edge from node i such that $w(i, j) \neq 0$, this directly implies that the denominator is nonzero:

$$\sum_{k \in D(i)} |w(i, k)| > 0$$

. Indeed, the existence of at least one non-zero weight $w(i, j)$ ensures that $|w(i, j)| > 0$, thus the sum of absolute values in the denominator must be strictly greater than zero.

Consequently, under conditions where $w(i, j) \neq 0$, both formulations are equivalent, as both have identical numerators and denominators. Furthermore, when $w(i, j) = 0$, both assign a normalized weight of 0. Hence, the two formulations are completely consistent with each other and yield identical normalized weight matrices. □

One could think that normalizing the weights based on the upstream edges doesn't reflect the true values of the original weights present in the graph, and that would be a good remark. That is why the framework also deploys an option to not normalize the weights in the graphs.

Notational Conventions Throughout the rest of this section, we will adopt the following:

- Bold lowercase (e.g., $\vec{v}$) for vectors representing node states.
- Bold uppercase (e.g., $\mathbf{W}$) for matrices.

- Subscripts (Ag) to denote association with an agent or its augmented graph.
- Superscripts (n) to denote iteration time.
- Operators $D(u, \mathcal{G})$, $U(v, \mathcal{G})$ for downstream and upstream neighborhoods.

For general background on graph types and structural metrics, see Appendix A.2.

A.3.2 Perturbation Propagation Model

In the MASFENON framework, the behavior of each agent is governed by a propagation model defined on its augmented graph $\mathcal{G}_{Ag}$. The perturbation vector $\vec{v}^{(n)}(\mathcal{G}_{Ag})$ encodes the activity or state of all nodes, including both intrinsic (gene-like) and virtual communication nodes, at iteration n .

Static Linear Model (Single-Step) The basic (non-iterative) propagation model assumes a single perturbation input defined initially at each node. Formally, given a perturbation input value for each node $u \in V(\mathcal{G}_{Ag})$, denoted by $Input_u$, the perturbation state at node $v \in V(\mathcal{G}_{Ag})$ after a single propagation step, $v_v(\mathcal{G}_{Ag})$, can be expressed as:

$$v_i(\mathcal{G}_{Ag}) = Input_i + \sum_{u \in U(i, \mathcal{G}_{Ag})} \bar{w}_{u,i}(\mathcal{G}_{Ag}) v_u(\mathcal{G}_{Ag}) \quad (\text{A.1})$$

where $U(v, \mathcal{G}_{Ag})$ is the set of upstream neighbors of node v , and $\bar{w}_{u,v}(\mathcal{G}_{Ag})$ is the normalized weight of the edge from node u to node v .

This node-level equation indicates that the perturbation at node v is determined by its direct perturbation input $Input_v$ plus the weighted sum of perturbations propagated from its upstream neighbors.

Aggregating these node-level equations for all nodes results in the compact matrix representation:

$$(I - \bar{W}(\mathcal{G}_{Ag})^T) \vec{v} = \vec{Input}, \quad (\text{A.2})$$

where I is the identity matrix, and $\vec{v}$, $\vec{Input}$, and $\bar{W}(\mathcal{G}_{Ag})$ are defined as before. The proof for the passage from the node-level equation to the matrix representation is the following:

Proof. Starting from the matrix-based formulation:

$$(I - \bar{W}(\mathcal{G}_{Ag})^T) \vec{v} = \vec{Input}$$

Expanding the terms explicitly, we obtain:

$$\vec{v} - \bar{W}(\mathcal{G}_{Ag})^T \vec{v} = \vec{Input}$$

Considering each component $v_i(\mathcal{G}_{Ag})$ separately, we have:

$$v_i(\mathcal{G}_{Ag}) - \sum_{j \in V(\mathcal{G}_{Ag})} \bar{w}_{j,i}(\mathcal{G}_{Ag}) v_j(\mathcal{G}_{Ag}) = Input_i$$

We recall that the normalized weight $\bar{w}_{j,i}(\mathcal{G}_{Ag})$ is nonzero only if there exists an edge from node j to node i , otherwise it is zero. Hence, we simplify the summation to only consider upstream neighbors of node i :

$$v_i(\mathcal{G}_{Ag}) - \sum_{j \in U(i, \mathcal{G}_{Ag})} \bar{w}_{j,i}(\mathcal{G}_{Ag}) v_j(\mathcal{G}_{Ag}) = Input_i$$

Rearranging terms, we obtain exactly the node-level formulation (Eq. A.1):

$$v_i(\mathcal{G}_{Ag}) = Input_i + \sum_{j \in U(i, \mathcal{G}_{Ag})} \bar{w}_{j,i}(\mathcal{G}_{Ag}) v_j(\mathcal{G}_{Ag})$$

This demonstrates that the matrix formulation and node-level equation represent the same linear system of perturbation propagation. $\square$

This linear system can be solved using the Moore-Penrose pseudo-inverse:

$$\vec{v} = (I - \bar{W}^T)^+ Input,$$

which provides a least-squares solution even when the system is underdetermined or non-invertible. As we have discussed before, the Moore-Penrose is not stable and can quickly lead to unforeseen behavior, so the use of this function to propagate should be limited or used under the assumptions that it could lead to unstable results.

Recursive Model (Iterative Dynamics) To simulate progressive perturbation dynamics over discrete time steps, MASFENON defines a recursive propagation rule. Let $\vec{v}^{(n)}$ denote the state at time n , and f the propagation function. The update rule is:

$$\vec{v}^{(n)} = f(\mathcal{G}, \vec{v}^{(n-1)}, t_n) = \vec{v}^{(n-1)} + \omega(t_n) \bar{W}(\mathcal{G})^T \vec{v}^{(n-1)}, \quad (\text{A.3})$$

where:

- $\omega(t_n)$ is a user-defined propagation scaling function.
- $\bar{W}(\mathcal{G})^T \vec{v}^{(n-1)}$ propagates influence from upstream nodes.

Recursive Node-Level Form The model can also be written in node-wise recursive form:

$$v_i^{(n)} = v_i^{(n-1)} + \omega(t_n) \sum_{u \in U(i, \mathcal{G})} \bar{w}_{u,i} \cdot v_u^{(n-1)}.$$

This explicitly expresses each node's new state as a combination of its previous state and the weighted contributions from upstream neighbors.

Initial Conditions At iteration $n = 0$, the initial input vector is used:

$$\vec{v}^{(0)} = \vec{Input}.$$

Iterative vs. Closed-Form Equation (A.2) defines a closed-form solution for a one-step system. For dynamic propagation, the recursive formulation is preferred, particularly in multi-agent contexts where interactions (inter-agent propagation) occur intermittently between intra-agent propagation cycles.

Interpretation This propagation model simulates local perturbation diffusion through the intra-agent network, analogous to information spread or signal transduction. The weights $\bar{w}_{ij}$ capture directional influence and are normalized to ensure numerical stability and comparability across nodes of varying degree or strength.

Extension Subsequent subsections describe how this core model is augmented with:

- **Dissipation** functions, which model loss of signal strength over time;
- **Conservation** mechanisms, which preserve perturbation under network constraints;
- **Inter-agent communication**, which synchronizes virtual node values between agents.

A.3.3 Dissipation and Conservation Dynamics

Realistic simulation of agent behavior in complex systems often requires that propagation dynamics account for constraints such as decay of influence and energy conservation. MASFENON introduces two mathematical constructs to this end: a *dissipation function* that models the temporal decay of perturbation, and a *conservation function* that ensures local or global preservation of perturbation across iterations.

Dissipation Function The dissipation mechanism modulates the perturbation values before they are used in propagation. This simulates natural decay, signal loss, or forgetfulness across time. Given a perturbation vector $\vec{v} \in \mathbb{R}^n$, the dissipation function is defined as:

$$d(\vec{v}, t_n) = \vec{v} - \lambda(t_n)\vec{v} = (1 - \lambda(t_n))\vec{v}, \quad (\text{A.4})$$

where:

- $\lambda : \mathbb{R} \rightarrow [0, 1]$ is a user-defined dissipation rate function;
- t_n is the global time (or iteration step).

Larger $\lambda(t_n)$ values induce stronger decay, while $\lambda(t_n) = 0$ results in no dissipation. This enables MASFENON to simulate both memory-preserving and forgetful systems.

Conservation Function To complement dissipation, MASFENON optionally enforces a form of perturbation conservation across the agent’s augmented network $\mathcal{G}$. Let $\overline{W}(\mathcal{G})$ be the normalized adjacency matrix and $\vec{q}(\mathcal{G})$ a scaling vector of node weights. The conservation function is:

$$c(\mathcal{G}, \vec{v}, t_n) = \theta(t_n) \cdot (\overline{W}(\mathcal{G})\vec{q}(\mathcal{G})) \odot \vec{v}, \quad (\text{A.5})$$

where:

- $\theta : \mathbb{R} \rightarrow \mathbb{R}$ is a conservation scaling function;
- $\odot$ denotes the Hadamard (element-wise) product;
- $\vec{q}(\mathcal{G})$ is user-defined (e.g., all ones for uniform conservation, or centrality-based);
- The term $\overline{W}\vec{q}$ captures local outflow per node.

Intuitively, conservation subtracts from each node the amount of signal it is expected to propagate outward to neighbors. This prevents unbounded accumulation of perturbation and promotes physically realistic behavior in closed systems.

Combined Correction in Iterative Dynamics The corrected perturbation value used in propagation is defined by applying both dissipation and conservation:

$$\vec{v}^{(n)} = f\left(\mathcal{G}, d(\vec{v}^{(n-1)}, t_n), t_n\right) - c\left(\mathcal{G}, d(\vec{v}^{(n-1)}, t_n), t_n\right), \quad (\text{A.6})$$

where f is the propagation function seen previously. This yields:

$$\vec{v}^{(n)} = d(\vec{v}^{(n-1)}, t_n) + \omega(t_n)\overline{W}^T d(\vec{v}^{(n-1)}, t_n) - c(\cdot),$$

which reflects the full intra-agent update at iteration n .

Practical Considerations Both $\lambda(t_n)$ and $\theta(t_n)$ can be fixed constants or dynamic functions (e.g., sinusoidal, decaying, or stochastic). This flexibility allows MASFENON to support diverse models of signal stability, conservation pressure, or feedback damping.

Relation to Physical Systems The dissipation term corresponds to friction or energy loss, while the conservation term resembles Kirchhoff-type flow balance. Together, they provide a general mechanism for simulating constrained dynamics over evolving state vectors in networked systems.

The next subsection extends this model to inter-agent propagation, where perturbation signals are transferred across agents using virtual input/output nodes synchronized at specific temporal intervals.

A.3.4 Inter-agent Communication Mechanism

In MASFENON, agents are modeled as independent entities with their own internal network structures. Communication between these agents is governed by a temporal multi-graph $\text{Agents} = (V(\text{Agents}), E(\text{Agents}))$, where vertices represent agents and edges encode interaction events annotated by contact times.

To simulate cross-agent communication, each agent is assigned a set of *virtual input* and *virtual output* nodes that bridge their internal dynamics with other agents. The values of these virtual nodes are synchronized based on inter-agent contact schedules, forming the backbone of the multi-agent simulation.

Virtual Node Assignment For every agent $Ag \in V(\text{Agents})$, the augmented graph $\mathcal{G}_{Ag} = (V, E)$ includes:

- A set of input nodes $V_{\text{in}}(Ag)$ receiving information from other agents;
- A set of output nodes $V_{\text{out}}(Ag)$ broadcasting internal signals to others;
- Edges $E_{\text{in}}(Ag)$ and $E_{\text{out}}(Ag)$ connecting virtual nodes to internal network nodes;
- The augmented node set:

$$V(\mathcal{G}_{Ag}) = V(G_{Ag}) \cup V_{\text{in}}(Ag) \cup V_{\text{out}}(Ag).$$

Each virtual node pair $(v_{Ag_s}^{\text{out}}, v_{Ag_t}^{\text{in}})$ represents a communication link from agent Ag_s to Ag_t .

Inter-agent Synchronization Rule Inter-agent communication occurs every N_{intra} intra-iterations, corresponding to a fixed timestep duration. At each inter-propagation iteration $n = k \cdot N_{\text{intra}}$, virtual input nodes are updated according to:

$$\vec{v}_{v_{\text{in}}}^{(n)}(\mathcal{G}_{Ag_t}) = \vec{v}_{v_{\text{out}}}^{(n)}(\mathcal{G}_{Ag_s}), \quad (\text{A.7})$$

where:

- $v^{\text{in}} \in V_{\text{in}}(Ag_t)$ is the receiving virtual node in the target agent;
- $v^{\text{out}} \in V_{\text{out}}(Ag_s)$ is the broadcasting virtual node in the source agent;
- The mapping between them is determined by edges $(Ag_s, Ag_t, t) \in E(\text{Agents})$.

If multiple contact events are detected between Ag_s and Ag_t within a timestep, their cumulative effect is modeled by either:

1. Repeating the propagation multiple times (multiplicative strength);
2. Averaging or summing the outputs (depending on simulation mode).

Temporal Quantization To facilitate temporal modeling, contact times are quantized into discrete windows $[t_i, t_{i+1})$, each associated with a fixed number of intra-iterations. The function $\#contacts(t_i, t_{i+1})$ counts interaction events within this interval, impacting how propagation strength is interpreted.

Integration into the Iterative Framework The inter-agent synchronization is invoked at the end of every N_{intra} internal iterations. During inter-propagation steps:

1. All virtual output values are collected from agents (local computation);
2. Values are exchanged via `acrshortmpi` (distributed communication);
3. Virtual inputs are updated in recipient agents (remote assimilation).

Mathematical Representation Given:

$$\vec{v}^{(n)}(\mathcal{G}_{Ag}) \in \mathbb{R}^{\dim(Ag)}, \quad Ag \in V(\text{Agents}),$$

the update of the state vector at inter-step n incorporates the remote perturbation as:

$$\vec{v}^{(n)}(\mathcal{G}_{Ag_t})[v^{\text{in}}] \leftarrow \vec{v}^{(n)}(\mathcal{G}_{Ag_s})[v^{\text{out}}], \quad (\text{A.8})$$

propagated forward in the next intra-propagation round using the same dynamics defined in Section A.3.2.

This structured communication strategy enables MASFENON to accurately simulate spatiotemporal information flow in multi-agent networks, allowing perturbation to travel not only within graphs but also across them. It supports applications from epidemiology to distributed signaling in cellular systems.

A.3.5 Correction Strategies: Normalization, Dissipation, Conservation

In dynamic multi-agent simulations, repeated propagation may lead to runaway values or convergence artifacts. To mitigate these effects and enforce biologically or physically plausible dynamics, MASFENON incorporates three correction strategies, each applied after intra- or inter-propagation steps.

These correction functions are designed to operate on the evolving state vector $\vec{v}^{(n)}(\mathcal{G})$, and act either before or after applying the main propagation kernel. Below, we define and formalize each correction mechanism.

Normalization / Saturation via Scaled Tanh

Propagation values may exceed biologically realistic bounds if left unchecked. To ensure that the signal magnitude remains interpretable, MASFENON applies a saturating non-linear transformation using a scaled hyperbolic tangent function:

$$\text{scaledTanh}(x, \gamma) = \gamma \cdot \tanh\left(\frac{x}{\gamma}\right) \quad (\text{A.9})$$

This function rescales inputs to the range $[-\gamma, \gamma]$, enforcing bounded growth. Here, the parameter $\gamma > 0$ controls the degree of saturation. As $x \rightarrow \pm\infty$, the function asymptotically approaches $\pm\gamma$, enforcing bounded perturbation values. It is applied element-wise to the post-propagation vector:

$$\vec{v}^{(n)} \leftarrow \text{scaledTanh}(\vec{v}^{(n)}, \gamma)$$

where γ is a user-defined saturation parameter controlling the maximum allowed perturbation.

An alternative to the use of the scaled hyperbolic tangent function for value saturation is a thresholding (clipping) function. Both strategies are implemented in the framework and can be selected depending on the simulation requirements:

- **Scaled Hyperbolic Tangent (tanh):** This approach applies a continuous non-linear squashing function that asymptotically bounds the values within a specified interval. It integrates smoothly into the propagation formula and avoids discontinuities. However, it introduces slight value compression near the boundaries, meaning that values never exactly reach the saturation limits, which may result in minor dissipation of influence over time.
- **Thresholding (Clipping):** This method applies a hard limit to values, truncating any input that exceeds a defined threshold. It is straightforward to interpret and guarantees that the values remain strictly within bounds without altering values that lie within the threshold range. Although mathematically less smooth than *tanh*, it preserves the original signal more faithfully up to the saturation point and is often preferred for stability in bounded systems.

Dissipation: Decay over Time

To reflect natural decay or energy loss in biological and physical processes, a dissipation term is applied to each node after propagation:

$$d(\vec{v}, t_n) = \vec{v} - \lambda(t_n)\vec{v} \quad (\text{A.10})$$

Here, $\lambda : \mathbb{R} \rightarrow \mathbb{R}$ is a tunable function representing the dissipation rate at time t_n . Possible choices include:

- Constant rate: $\lambda(t_n) = \lambda_0$
- Oscillatory decay: $\lambda(t_n) = A \sin(\omega t_n + \phi)$
- Stochastic damping: $\lambda(t_n)$ sampled from $[-1, 1]$

The parameters serve different purposes. $\lambda(n) \in [0, 1]$ is a time-dependent dissipation factor. This formulation can be extended to include more elaborate, non-uniform dissipation by defining $\vec{\lambda}(n) \in \mathbb{R}^{|V(\mathcal{G})|}$, allowing node-specific decay rates.

This term is applied after each propagation step to simulate signal degradation:

$$\vec{v}^{(n)} \leftarrow d(\vec{v}^{(n)}, t_n)$$

Conservation: Energy Balancing

In certain biological systems, perturbation propagation is constrained by conservation laws. MASFENON optionally applies a conservation correction to redistribute or reduce energy based on graph topology:

$$c(\mathcal{G}, \vec{v}, t_n) = \theta(t_n) \left((\overline{W}(\mathcal{G})\vec{q}) \odot \vec{v} \right) \quad (\text{A.11})$$

Where:

- $\theta(t_n) \in \mathbb{R}$: a time-dependent conservation is a scaling function (that will return the conservation coefficient).
- $\overline{W}(\mathcal{G})$: the row-normalized adjacency matrix.
- $\vec{q} \in \mathbb{R}^{|V|}$: an optional user-defined weight vector, encoding relative importance or outflow rate per node.
- $\odot$: element-wise (Hadamard) product.

This operation subtracts from each node's signal a portion proportional to how much it disseminates to its neighbors. The updated vector becomes:

$$\vec{v}^{(n)} \leftarrow \vec{v}^{(n)} - c(\mathcal{G}, \vec{v}^{(n)}, t_n)$$

Combined Correction Scheme

MAFENON optionally applies all three corrections in a pipeline, ensuring bounded, stable, and realistic propagation:

$$\vec{v}^{(n)} \leftarrow \text{scaledTanh} \left(d(\vec{v}^{(n)}, t_n) - c(\mathcal{G}, d(\vec{v}^{(n)}, t_n), t_n), \gamma \right)$$

The order of operations ensures:

1. **Dissipation** first reduces the signal;
2. **Conservation** adjusts it based on outflows;
3. **Normalization** compresses the output range.

Each correction mechanism is modular, enabling simulations to toggle components independently or in combination depending on the application domain (e.g., decay in cell signaling, conservation in resource flow).

Together, these strategies enhance the biological plausibility, mathematical stability, and computational robustness of the multi-agent propagation engine in MASFENON.

A.3.6 Iterative Propagation Algorithm

MAFENON operates via an iterative propagation procedure that alternates between intra-agent and inter-agent communication steps. The algorithm unfolds over discrete time iterations indexed by $n \in \mathbb{N}$, applying propagation, correction, and communication operators on the agent network and its internal graphs.

We denote the agent's augmented graph as $\mathcal{G}_{Ag} = (V(\mathcal{G}_{Ag}), E(\mathcal{G}_{Ag}))$, and its perturbation vector at step n as $\vec{v}^{(n)}(\mathcal{G}_{Ag})$.

The full update rule for intra-agent propagation is defined as:

$$\vec{v}^{(n)}(\mathcal{G}_{Ag}) = \begin{cases} \text{scaledTanh}(f(\vec{v}^{(0)}(\mathcal{G}_{Ag})), \gamma), & \text{if } n = 1 \\ \text{scaledTanh}(f(d(\vec{v}^{(n-1)}, t_n)) - c(\mathcal{G}_{Ag}, d(\vec{v}^{(n-1)}, t_n), t_n), \gamma), & \text{if } n > 1 \end{cases} \quad (\text{A.12})$$

Where:

- $\overline{W}(\mathcal{G}_{Ag})$ is the row-normalized adjacency matrix of $\mathcal{G}_{Ag}$;
- $d(\cdot, t_n)$ is the dissipation function (Eq. A.10);
- $c(\cdot, t_n)$ is the conservation function (Eq. A.11);
- $\text{scaledTanh}(\cdot, \gamma)$ is the saturation-normalization function (Eq. A.9);
- t_n is the physical time associated with iteration n , defined as $t_n = \frac{\text{timestep}}{N_{intra}} \cdot n$.

This rule updates perturbation values in each agent's local graph while preserving numerical stability and simulating realistic signal dynamics.

Inter-Agent Communication After every N_{intra} intra-iterations, agents exchange information via their virtual output and input nodes. For each edge $(Ag_i, Ag_j, d) \in E(Agents)$, communication occurs if the contact time d lies in the interval $[t_n, t_{n+1})$.

Let $v_{out} \in V_{out}(\mathcal{G}_{Ag_i})$ and $v_{in} \in V_{in}(\mathcal{G}_{Ag_j})$ be the corresponding communication nodes. Then we set:

$$\vec{v}_{v_{in}}^{(n)}(\mathcal{G}_{Ag_j}) \leftarrow \vec{v}_{v_{out}}^{(n)}(\mathcal{G}_{Ag_i}) \quad (\text{A.13})$$

This inter-agent exchange allows downstream propagation in the receiving network to incorporate the externally sourced information in the next intra-cycle.

Control Parameters The algorithm is governed by four main parameters:

- N_{inter} : number of inter-iterations (external communication rounds);
- N_{intra} : number of intra-iterations between inter-communications;
- $timestep$: physical time between two inter-iterations;
- γ : scaling bound for tanh or thresholding normalization.

Total iterations are given by $N = N_{inter} \cdot N_{intra}$. The current simulation time at iteration n is:

$$t_n = \frac{timestep}{N_{intra}} \cdot n$$

Pseudocode Overview For clarity, the high-level pseudocode of the full propagation algorithm is:

```

Initialize all node values  $\hat{v}(0)$ 
for inter_iter = 1 to  $N_{inter}$ :
    for intra_iter = 1 to  $N_{intra}$ :
        for each agent Ag:
            propagate the values inside the network of agent Ag
        perform inter-agent propagation via virtual nodes

```

Each round of intra-agent propagation is computationally intensive but fully parallelizable. The inter-agent step is communication-bound and implemented with MPI-based messaging.

This formal algorithm underpins the iterative engine of MASFENON and embodies its capacity to simulate information flow, perturbation dynamics, and multi-scale interactions across agents and their internal networks.

A.3.7 Complexity Analysis

The complexity of the MASFENON framework is driven primarily by the linear algebra operations performed during intra-agent propagation and the communication overhead during inter-agent interactions. This subsection analyzes both time and space complexity in relation to key structural and computational parameters.

Notation Let:

- $n = |V(\mathcal{G}_{Ag})|$: number of nodes (including virtual) in a single agent's augmented graph;
- $m = |E(\mathcal{G}_{Ag})|$: number of edges in the same graph;
- $A = |V(Agents)|$: number of agents;
- $C = |E(Agents)|$: number of inter-agent communication links;
- $N = N_{inter} \cdot N_{intra}$: total number of iterations;

Intra-Agent Time Complexity

The dominant operation per iteration is the matrix-vector product:

$$\vec{v}^{(n)} = \overline{W}^T \vec{v}^{(n-1)}$$

This has time complexity:

$$\mathcal{O}(m) \quad (\text{sparse adjacency matrices})$$

When performed across A agents, each operating on its own augmented graph, the per-iteration cost becomes:

$$\mathcal{O}(A \cdot m)$$

This is repeated N times, giving total intra-agent propagation cost:

$$\boxed{\mathcal{O}(A \cdot m \cdot N)}$$

If $\overline{W}$ is dense (e.g., in fully connected or highly regular graphs), the per-agent cost becomes $\mathcal{O}(n^2)$, and the total intra cost grows to:

$$\mathcal{O}(A \cdot n^2 \cdot N)$$

Inter-Agent Time Complexity

Every N_{intra} steps, inter-agent communication updates values in virtual input nodes from virtual output values in other agents. The number of updates is proportional to the number of active edges in the temporal agent network $E(Agents)$. Each communication is a constant-time update:

$$\vec{v}_{in}^{(n)} \leftarrow \vec{v}_{out}^{(n)}$$

Hence, the cost per communication round is:

$$\mathcal{O}(C)$$

Total number of such updates is:

$$N_{inter} \cdot C = \frac{N}{N_{intra}} \cdot C$$

So, the total inter-agent communication cost is:

$$\boxed{\mathcal{O}\left(C \cdot \frac{N}{N_{intra}}\right)}$$

Memory (Space) Complexity

Each agent stores its own:

- Adjacency matrix $\overline{W} \in \mathbb{R}^{n \times n}$, ideally sparse;
- Perturbation vector $\vec{v} \in \mathbb{R}^n$;
- Auxiliary vectors for dissipation and conservation;

In the worst case (dense matrices), per-agent memory usage is:

$$\mathcal{O}(n^2)$$

Across all agents:

$$\boxed{\mathcal{O}(A \cdot n^2)}$$

This complexity is in line with what was estimated in the main text in equation 3.15

Scalability Considerations

MASFENON’s performance is governed by the balance between:

- **Computation-bound** intra-agent matrix operations (highly parallelizable);
- **Communication-bound** inter-agent data exchange (dependent on network latency and bandwidth).

Optimal performance is achieved when:

$$\text{Computation Time} \gg \text{Communication Time}$$

which can be enforced via:

- Increasing N_{intra} relative to N_{inter} ;
- Using fast interconnects (e.g., Infiniband in MPI clusters);
- Co-locating agents with high interaction frequency on the same node.

Summary of Complexity

Table A.1: Complexity Summary of MASFENON Iterative Engine

Aspect	Complexity
Intra-agent time (sparse)	$\mathcal{O}(A \cdot m \cdot N)$
Intra-agent time (dense)	$\mathcal{O}(A \cdot n^2 \cdot N)$
Inter-agent communication	$\mathcal{O}(C \cdot \frac{N}{N_{intra}})$
Memory (dense)	$\mathcal{O}(A \cdot n^2)$

This analysis provides a theoretical grounding for the runtime performance observed in large-scale simulation experiments and guides system tuning for optimal throughput and resource allocation. This chapter was also in line

with what we have presented in the main text in section 4.1 and chapter 3, when discussing about complexity and confronted some experimental data with our assumptions.

A.3.8 Temporal Modeling and Inter/Intra Decomposition

In the MASFENON framework, agent-based dynamics are decomposed into two levels of propagation:

- **Intra-agent propagation**, local evolution of node states within the augmented graph $\mathcal{G}_{Ag}$, occurring over short time steps;
- **Inter-agent propagation**, transfer of information between agents through virtual node updates, occurring at larger time intervals.

Discrete Temporal Domain. Let the global temporal axis be discretized into uniform intervals of length $\Delta t \in \mathbb{R}^+$, with total time modeled as:

$$T = \{t_0, t_1, \dots, t_N\}, \quad t_n = t_0 + n \cdot \Delta t.$$

Each t_n corresponds to iteration n of the model.

Define two nested periodicities:

- $N_{\text{intra}} \in \mathbb{N}$: number of intra-iterations per inter-propagation step;
- $N_{\text{inter}} \in \mathbb{N}$: total number of inter-iteration steps;
- $N = N_{\text{intra}} \cdot N_{\text{inter}}$: total number of iterations.

Let:

$$\delta t = \frac{\Delta t}{N_{\text{intra}}}$$

be the *intra-step resolution*. Then iteration time becomes:

$$t_n = n \cdot \delta t.$$

Intra-Agent Dynamics. The intra-propagation is defined by the iterative update:

$$\vec{v}^{(n)}(\mathcal{G}) = f(\mathcal{G}, d(\vec{v}^{(n-1)}, t_n), t_n) - c(\mathcal{G}, d(\vec{v}^{(n-1)}, t_n), t_n)$$

where:

- f : propagation function (e.g., matrix product with normalized adjacency);
- d : dissipation function (time-modulated decay);
- c : conservation function (energy/perturbation leakage control).

Inter-Agent Synchronization. Let τ_k be the time step corresponding to the k^{th} inter-propagation event:

$$\tau_k = k \cdot N_{\text{intra}}, \quad k \in \{0, 1, \dots, N_{\text{inter}} - 1\}.$$

At each τ_k , the inter-agent network $Agents = (V, E)$ defines active communication links based on contact time. Each active edge $(Ag_i, Ag_j, d) \in E$ with $d \in [\tau_k \cdot \delta t, (\tau_k + N_{\text{intra}}) \cdot \delta t)$ triggers a virtual node update:

$$v_{in}^{(\tau_k)}(\mathcal{G}_{Ag_j}) \leftarrow v_{out}^{(\tau_k)}(\mathcal{G}_{Ag_i})$$

The value transfer ensures inter-agent coupling. In vector notation, define:

$$\forall (Ag_i, Ag_j, d) \in E, \quad \vec{v}_{in}^{(\tau_k)}(Ag_j) = \Pi_{Ag_i \rightarrow Ag_j} \vec{v}_{out}^{(\tau_k)}(Ag_i)$$

where Π is a linear projection operator aligning source outputs to target inputs.

Temporal Decomposition Summary. We define the total model as:

$$\forall n \in \{0, \dots, N-1\}, \quad \begin{cases} \textbf{Intra-step:} & \vec{v}^{(n+1)} \leftarrow f - c \circ d(\vec{v}^{(n)}, t_n) \\ \textbf{Inter-step:} & \text{If } n \bmod N_{\text{intra}} = 0 \text{ then update } \vec{v}_{in} \end{cases}$$

This decomposition allows flexible tuning of:

- Temporal resolution of local vs global propagation;
- Frequency of inter-agent synchronization;
- Separation of timescales in agent interaction vs internal network dynamics.

Time Complexity Refinement. This dual-scale time axis implies the following effective complexity:

- $N \cdot \mathcal{O}(n^2)$: intra-step complexity per agent (matrix operations);
- $\frac{N}{N_{\text{intra}}} \cdot \mathcal{O}(d_{\text{avg}} \cdot A)$: inter-step total update (assuming average out-degree d_{avg});

This decomposition also suggests optimization opportunities for asynchronous, event-driven, or coarse-grained scheduling in future implementations.

Temporal Quantization Strategies. Given a temporal multi-graph of agent interactions:

$$Agents = (V, E), \quad E \subseteq V \times V \times \mathbb{R},$$

where each edge $e = (Ag_i, Ag_j, t)$ represents a contact at real-valued time $t \in \mathbb{R}$, we must map continuous contact times into discrete inter-iteration windows:

$$[t_k, t_{k+1}) = [k \cdot N_{\text{intra}} \cdot \delta t, (k+1) \cdot N_{\text{intra}} \cdot \delta t).$$

Let $\mathcal{C}_k(Ag_i, Ag_j) \subseteq \mathbb{R}$ be the set of real contact times falling into interval $[t_k, t_{k+1})$ between agent Ag_i and Ag_j .

Contact Quantization Operator. Define the quantized contact function:

$$\#contacts_{[t_k, t_{k+1})}(Ag_i, Ag_j) = |\mathcal{C}_k(Ag_i, Ag_j)|.$$

This value supports two formal quantization strategies:

(Q1) Boolean Presence Strategy (Contact Indicator). Define the binary indicator function:

$$\chi_k(Ag_i, Ag_j) = \begin{cases} 1 & \text{if } |\mathcal{C}_k(Ag_i, Ag_j)| > 0, \\ 0 & \text{otherwise.} \end{cases}$$

This is used to activate a unit-weighted propagation whenever any interaction is detected in the window.

(Q2) Contact Weight Accumulation. Define the weighted contact multiplicity:

$$\eta_k(Ag_i, Ag_j) = |\mathcal{C}_k(Ag_i, Ag_j)| \in \mathbb{N}.$$

This scalar modifies the strength of the virtual node update during inter-propagation. Let $\Pi^{(\eta)}$ be a scaled projection operator such that:

$$\vec{v}_{in}^{(n)}(Ag_j) = \Pi_{Ag_i \rightarrow Ag_j}^{(\eta_k)} \vec{v}_{out}^{(n)}(Ag_i),$$

with:

$$\Pi_{Ag_i \rightarrow Ag_j}^{(\eta_k)} = \eta_k(Ag_i, Ag_j) \cdot \Pi_{Ag_i \rightarrow Ag_j}.$$

Modeling Notes.

- Strategy Q1 enforces binary logic (contact or no contact) and is robust to noise.
- Strategy Q2 encodes contact density, enabling a weighted information flow between agents, and better reflects bursty or repeated interactions.

Matrix Formulation. Let $\mathcal{Q}_k \in \mathbb{R}^{|V| \times |V|}$ be the quantized contact matrix at inter-iteration k , defined by:

$$\mathcal{Q}_k[Ag_i, Ag_j] = \begin{cases} \chi_k(Ag_i, Ag_j) & \text{in Q1,} \\ \eta_k(Ag_i, Ag_j) & \text{in Q2.} \end{cases}$$

Then the inter-agent propagation operation over all agents becomes:

$$\vec{V}_{in}^{(n)} = \mathcal{Q}_k \odot \Pi \vec{V}_{out}^{(n)},$$

where $\odot$ is applied entry-wise across agent pairs and Π is the base inter-agent projection operator matrix.

Future Extensions. MASFENON’s formulation admits:

- *Stochastic quantization* with Poisson-distributed contact modeling;
- *Temporal smoothing*, using Gaussian kernels over contact density;
- *Edge attribute filtering*, where η_k is modulated by edge weight functions or confidence scores.

A.3.9 Inter-Agent Communication Modeling

A critical component of the MASFENON framework is its ability to simulate communication between agents over time. This is achieved through a formal inter-agent propagation mechanism using **virtual nodes** and **projection operators**. These operators serve as mathematical mappings to transfer perturbation values from the virtual output nodes of one agent to the virtual input nodes of another, during temporally quantized interaction events.

Let $\text{Agents} = (V(\text{Agents}), E(\text{Agents}))$ denote the inter-agent temporal network, and let $\mathcal{G}_{Ag_i}$ be the augmented graph associated with agent Ag_i . Each such graph contains standard graph nodes and two sets of virtual nodes:

- $V_{out}(Ag_i) \subset V(\mathcal{G}_{Ag_i})$: virtual nodes representing output interfaces for inter-agent propagation.
- $V_{in}(Ag_j) \subset V(\mathcal{G}_{Ag_j})$: virtual nodes representing input interfaces to receive propagated states from others.

We define the **projection operator** $\Pi_{Ag_i \rightarrow Ag_j} \in \mathbb{R}^{\dim(Ag_j) \times \dim(Ag_i)}$ as a sparse matrix that maps output node values of Ag_i into the corresponding input positions of Ag_j . This operator is zero everywhere except for entries connecting matched virtual output/input nodes across agents.

Unweighted inter-propagation (Q1 strategy):

$$\vec{v}_{in}^{(n)}(Ag_j) = \Pi_{Ag_i \rightarrow Ag_j} \vec{v}_{out}^{(n)}(Ag_i)$$

Weighted inter-propagation (Q2 strategy): Let $\eta_k(Ag_i, Ag_j) \in \mathbb{N}$ denote the number of observed contact events between agents Ag_i and Ag_j during interval k . Then:

$$\vec{v}_{in}^{(n)}(Ag_j) = \eta_k(Ag_i, Ag_j) \cdot \Pi_{Ag_i \rightarrow Ag_j} \vec{v}_{out}^{(n)}(Ag_i)$$

This modeling allows fine control over:

- the *topology* of communication across agents (via projection structure),
- the *dynamics* of inter-agent propagation (via iteration-dependent values),
- the *intensity* of interaction (via quantized weights η_k).

Composition and propagation: If multiple agents are transmitting to Ag_j during a timestep, the total virtual input vector is the sum over all projections:

$$\vec{v}_{in}^{(n)}(Ag_j) = \sum_{Ag_i \in \mathcal{N}_{in}(Ag_j)} \eta_k(Ag_i, Ag_j) \cdot \Pi_{Ag_i \rightarrow Ag_j} \vec{v}_{out}^{(n)}(Ag_i)$$

Here, $\mathcal{N}_{in}(Ag_j)$ denotes the set of agents with an outgoing edge to Ag_j in the inter-agent temporal graph.

This inter-agent projection framework enables MASFENON to simulate dynamic, temporally-quantized flows of perturbation through a population of network-embedded agents, bridging heterogeneous topologies and asynchronous communication patterns.

A.3.10 Robustness Mechanisms and Correction Models

To ensure stable and interpretable propagation within and across agent-associated networks, the MASFENON framework incorporates several mechanisms designed to enhance robustness, mitigate numerical instability, and reflect biologically or structurally plausible behaviors. These mechanisms, namely, **normalization**, **dissipation**, and **conservation**, act as correction models integrated into the iterative propagation dynamics.

1. Saturation and Normalization via Scaled Hyperbolic Tangent

The scaled hyperbolic tangent function seen in eq. A.9 is used to constrain the range of node values during propagation. This non-linearity avoids uncontrolled value escalation, simulating saturation effects observed in real biological or dynamical systems.

This function is applied element-wise to the propagated state vector after each iteration, ensuring that even in the presence of positive feedback loops, the system does not diverge numerically. Although the scaled hyperbolic tangent leads to a sound modeling of the saturation, it is usually preferred to use the thresholding since using the tangent also dissipates the values, so it tends to be destructive on values that could be modeled as non-changing during the times when the values are under a threshold (e.g. epidemics and testing the behavior in some closed environments).

2. Dissipation Dynamics Dissipation reflects a decay or leakage of influence over time, often modeling biological degradation or energy loss. It is implemented via a linear contraction on the propagated vector, as seen previously in eq. A.10. There could be also other ways of implementing the dissipation, with more constraints and interactions between the different nodes in a network, but we want to keep these dynamics really simple,

since the complexity and the modeling of complex systems does not need to be using some complex frameworks with complex dynamics. On the contrary, some very complex behavior can be simulated by very simple rules and models.

3. Conservation Effects The conservation mechanism introduces a penalization term representing the amount of influence a node redistributes to its neighbors, based on the outgoing edges as seen in eq. A.11. This formulation reflects the assumption that sending information to neighbors comes at the cost of decreasing one’s own value, capturing conservation constraints.

4. Combined Correction Model The complete corrected state update for the network associated with an agent Ag is defined recursively as:

$$\vec{v}^{(n)}(\mathcal{G}_{Ag}) = \text{scaledTanh} \left(\overline{W}(\mathcal{G}_{Ag})^\top \cdot \vec{v}_{\text{diss}}^{(n-1)} - \theta(n) \cdot \left(\overline{W}(\mathcal{G}_{Ag}) \vec{q} \odot \vec{v}_{\text{diss}}^{(n-1)} \right), \gamma \right) \quad (\text{A.14})$$

where:

$$\vec{v}_{\text{diss}}^{(n-1)} = \vec{v}^{(n-1)} - \lambda(n) \cdot \vec{v}^{(n-1)}$$

This compact formulation merges propagation, saturation, dissipation, and conservation into a unified iteration scheme. It is stable under bounded $\lambda(n)$, $\theta(n)$, and γ , and allows flexible extension for agent-level heterogeneity via node-wise parameter vectors.

5. Observations and Constraints To ensure robustness and convergence:

- The spectral radius of $\overline{W}$ should be < 1 , or propagation should be combined with bounded non-linearities.
- Functions $\lambda(n)$ and $\theta(n)$ can be constant or adaptive (e.g., decreasing, periodic).
- The vector $\vec{q}$ can encode structural roles, random noise, or domain-informed constraints.

These mechanisms collectively define the correction layer that transforms linear propagation into a robust, biologically and computationally grounded iterative process.

A.3.11 Convergence Analysis and Stability Conditions

To ensure that the perturbation propagation algorithm defined by the MAS-FENON framework behaves predictably over multiple iterations, we must examine the conditions under which the system converges, remains bounded, or diverges.

Let us consider the corrected propagation model (using neighborhood propagation) as defined in Eq. A.14:

$$\vec{v}^{(n)} = \text{scaledTanh} \left(\overline{W}^\top \vec{v}_{\text{diss}}^{(n-1)} - \theta(n)(\overline{W} \vec{q} \odot \vec{v}_{\text{diss}}^{(n-1)}), \gamma \right),$$

where:

$$\vec{v}_{\text{diss}}^{(n-1)} = \vec{v}^{(n-1)} - \lambda(n)\vec{v}^{(n-1)}.$$

We investigate convergence in both the **linearized** case and in the presence of **nonlinear corrections** (saturation, dissipation, conservation).

1. Linear Baseline Convergence In the absence of correction mechanisms ($\lambda = 0$, $\theta = 0$, $\text{scaledTanh} \approx \text{identity}$), the update rule simplifies to:

$$\vec{v}^{(n)} = \overline{W}^\top \vec{v}^{(n-1)}.$$

This is a classic linear iteration whose convergence depends on the spectral radius $\rho(\overline{W}^\top)$:

Theorem A.3.1 (Linear Convergence Criterion). *The iteration*

$$\vec{v}^{(n)} = \overline{W}^\top \vec{v}^{(n-1)}$$

converges to the zero vector if and only if $\rho(\overline{W}) < 1$, where $\rho(\cdot)$ denotes the spectral radius.

If $\rho(\overline{W}) = 1$, the system may converge, oscillate, or diverge depending on the structure of the eigenvectors. If $\rho(\overline{W}) > 1$, divergence is guaranteed.

2. Influence of Dissipation When $\lambda(n) > 0$, each iteration applies an additional contraction:

$$\vec{v}^{(n)} = \overline{W}^\top [(1 - \lambda(n))\vec{v}^{(n-1)}].$$

This corresponds to a time-varying operator:

$$\vec{v}^{(n)} = \left((1 - \lambda(n))\overline{W}^\top \right) \vec{v}^{(n-1)}.$$

Convergence is now governed by:

$$\limsup_{n \rightarrow \infty} \left\| \prod_{i=1}^n (1 - \lambda(i)) \cdot \overline{W}^\top \right\| < 1.$$

Sufficient conditions include:

- $\rho(\overline{W}) < 1$, and
- $\lambda(n) \geq \lambda_0 > 0$ (constant or bounded below).

3. Influence of Saturation Applying the scaled hyperbolic tangent yields:

$$\vec{v}^{(n)} = \gamma \cdot \tanh \left(\text{linear expression in } \vec{v}^{(n-1)} \right),$$

which inherently constrains $\|\vec{v}^{(n)}\|_\infty \leq \gamma$ for all n . Thus, even if the linear system is unstable, the saturation guarantees *boundedness*, though not necessarily convergence.

This mechanism functions as a nonlinear projection onto a compact convex set $[-\gamma, \gamma]^d$. The iterates remain bounded, and may exhibit oscillatory or fixed-point behavior.

4. Conservation Coupling and Nonlinear Feedback The conservation term introduces an implicit feedback of the form:

negative feedback proportional to outflow,

which may increase contraction in certain regions of the state space.

The total update then becomes a composition of: - A contraction (if $\rho(\bar{W}) < 1$), - A damping (via $\lambda(n)$), - A nonlinear saturation (via $\tanh$), - A negative feedback term (via conservation).

This composition ensures convergence to a stable fixed point under broad conditions.

Theorem A.3.2 (Sufficient Condition for Convergence). *Assume the following:*

- The spectral radius $\rho(\bar{W}) < 1$.
- The dissipation factor $\lambda(n) \geq \lambda_0 > 0$ uniformly.
- The conservation factor $\theta(n) \leq \theta_0$ bounded.
- The saturation function scaled $\text{Tanh}(x, \gamma)$ is applied at every iteration.

Then the sequence $\vec{v}^{(n)}$ converges to a bounded fixed point.

5. Practical Remarks

- **Oscillatory regimes** can arise in the absence of saturation and dissipation, especially when $\bar{W}$ has complex eigenvalues.
- **Pseudo-inverse instabilities** (in early formulations using $(I - \bar{W}^\top)^+$) can be avoided by checking matrix conditioning and applying regularization.
- **Parameter tuning** for γ , $\lambda(n)$, and $\theta(n)$ is essential for balancing convergence speed and expressive dynamics.

In all cases, careful selection of the correction parameters ensures a mathematically grounded and computationally robust evolution of network states.

A.3.12 Numerical Stability of the Pseudoinverse

In the MASFENON framework, the use of the Moore-Penrose pseudoinverse $(I - \overline{W}^\top)^+$ arises naturally in solving systems involving the augmented adjacency matrix of each agent. While theoretically well-defined, this approach may suffer from **numerical instability** in practice, especially when the matrix $I - \overline{W}^\top$ is:

- Nearly singular or poorly conditioned,
- Large and sparse with inconsistent rank properties,
- Susceptible to rounding errors due to ill-scaled weights or precision limitations.

This instability can propagate across iterations, resulting in:

- Spurious or amplified values in the perturbation vector,
- Slow convergence or oscillatory behavior,
- Divergence under minor numerical perturbations.

To address this, the framework provides:

1. A **customizable propagation function** $f(\cdot)$ that users can replace or redefine according to stability requirements or domain-specific constraints.
2. A built-in **neighborhood propagation mode**, where only local node neighborhoods contribute to updates, bypassing matrix inversions altogether.

These alternatives offer greater control and numerical robustness in large-scale or ill-conditioned network scenarios. In particular, neighborhood-based propagation retains locality and sparsity, and is often preferable in real-time or HPC environments.

Recommendation. For high-dimensional simulations or real-world datasets with noisy or heterogeneous graph structures, avoid direct use of pseudoinverses unless the matrix is provably well-conditioned or regularized.

List of terms

C2C-SEPIA Cell to Cell Single-cell enriched pathway impact analysis. 8, 64, 78, 81, 134, 135, 197

MASFENON Multiagent Adaptive Simulation Framework for Evolution in Networks of Networks. 1–4, 13, 19, 20, 31, 34–36, 39, 41, 42, 49, 51, 52, 55, 57, 58, 60, 61, 63, 64, 66, 67, 69, 71–79, 81–84, 88–96, 98–126, 131, 133, 148, 151, 156, 159–167, 169–172

MPI Message Passing Interface. 2, 4, 52, 61, 63, 72, 74, 79, 103, 106–109, 111, 113–115, 124, 125, 133, 209, 211

scRNA-seq Single cell rna-sequencing. 2, 12, 13, 49, 72, 81, 82, 118, 134–136, 162

Bibliography

- [1] Salvatore Alaimo et al. “PHENSIM: phenotype Simulator”. In: *PLoS computational biology* 17.6 (2021), e1009069.
- [2] Norah Alghamdi et al. “A graph neural network model to estimate cell-wise metabolic flux using single-cell RNA-seq data”. In: *Genome research* 31.10 (2021), pp. 1867–1884.
- [3] Simon Anders, Paul Theodor Pyl, and Wolfgang Huber. “HTSeq—a Python framework to work with high-throughput sequencing data”. In: *bioinformatics* 31.2 (2015), pp. 166–169.
- [4] Erick Armingol et al. “Deciphering cell–cell interactions and communication from gene expression”. In: *Nature Reviews Genetics* 22.2 (2021), pp. 71–88.
- [5] Erick Armingol et al. “Deciphering cell–cell interactions and communication from gene expression”. In: *Nature Reviews Genetics* 22.2 (2021), pp. 71–88.
- [6] Albert-László Barabási and Réka Albert. “Emergence of scaling in random networks”. In: *science* 286.5439 (1999), pp. 509–512.
- [7] Fabio Belfemine, Agostino Poggi, and Giovanni Rimassa. “JADE–A FIPA-compliant agent framework”. In: *Proceedings of PAAM*. Vol. 99. 97-108. London. 1999, p. 33.
- [8] Volker Bergen et al. “Generalizing RNA velocity to transient cell states through dynamical modeling”. In: *Nature biotechnology* 38.12 (2020), pp. 1408–1414.
- [9] Jan Berkhout, Frank J Bruggeman, and Bas Teusink. “Optimality principles in the regulation of metabolic networks”. In: *Metabolites* 2.3 (2012), pp. 529–552.
- [10] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*. 1. Oxford university press, 1999.
- [11] Rafael H Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.

- [12] Fred Brauer. “Compartmental models in epidemiology”. In: *Mathematical epidemiology* (2008), pp. 19–79.
- [13] Lukas Breitwieser et al. “BioDynaMo: a modular platform for high-performance agent-based simulation”. In: *Bioinformatics* 38.2 (2022), pp. 453–460.
- [14] Robin Browaeys, Wouter Saelens, and Yvan Saeys. “NicheNet: modeling intercellular communication by linking ligands to target genes”. In: *Nature methods* 17.2 (2020), pp. 159–162.
- [15] Zixuan Cang and Qing Nie. “Inferring spatial and signaling relationships between cells from single cell transcriptomic data”. In: *Nature communications* 11.1 (2020), p. 2084.
- [16] Jiaji George Chen et al. “Giotto Suite: a multi-scale and technology-agnostic spatial multi-omics analysis ecosystem”. In: *Biorxiv* (2023).
- [17] Jiayi Chen et al. “Comparative RNA-Seq transcriptome analysis on silica induced pulmonary inflammation and fibrosis in mice silicosis model”. In: *Journal of Applied Toxicology* 38.5 (2018), pp. 773–782.
- [18] Jun Chen et al. “Spatial transcriptomic analysis of cryosectioned tissue samples with Geo-seq”. In: *Nature protocols* 12.3 (2017), pp. 566–580.
- [19] S Coakley et al. *Flexible large-scale agent modelling environment (flame). university of sheffield and rutherford appleton laboratories, stfc, license: Lesser gpl v3*. 2012.
- [20] Nick Collier. “Repast: An extensible framework for agent simulation”. In: *The University of Chicago’s social science research* 36 (2003), p. 2003.
- [21] Ilaria Cosentini et al. “Measuring cancer driving force of chromosomal aberrations through multi-layer Boolean implication networks”. In: *Plos one* 19.4 (2024), e0301591.
- [22] Kristopher De Asis et al. “Multi-step reinforcement learning: A unifying algorithm”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.
- [23] Scott De Marchi and Scott E Page. “Agent-based models”. In: *Annual Review of political science* 17.1 (2014), pp. 1–20.
- [24] David van Dijk et al. “MAGIC: A diffusion-based imputation method reveals gene-gene interactions in single-cell RNA-sequencing data”. In: *BioRxiv* (2017), p. 111591.
- [25] Daniel Dimitrov et al. “LIANA+ provides an all-in-one framework for cell-cell communication inference”. In: *Nature Cell Biology* 26.9 (2024), pp. 1613–1622.

- [26] Alexander Dobin et al. “STAR: ultrafast universal RNA-seq aligner”. In: *Bioinformatics* 29.1 (2013), pp. 15–21.
- [27] Ali Dorri, Salil S Kanhere, and Raja Jurdak. “Multi-agent systems: A survey”. In: *Ieee Access* 6 (2018), pp. 28573–28593.
- [28] Ali Dorri, Salil S. Kanhere, and Raja Jurdak. “Multi-Agent Systems: A Survey”. In: *IEEE Access* 6 (2018), pp. 28573–28593. DOI: 10.1109/ACCESS.2018.2831228.
- [29] Ruben Dries et al. “Giotto: a toolbox for integrative analysis and visualization of spatial expression data”. In: *Genome biology* 22 (2021), pp. 1–31.
- [30] Elias Farr et al. “MetalinksDB: a flexible and contextualizable resource of metabolite-protein interactions”. In: *Briefings in Bioinformatics* 25.4 (2024), bbae347.
- [31] Tim Finin et al. “KQML as an agent communication language”. In: *Proceedings of the third international conference on Information and knowledge management*. 1994, pp. 456–463.
- [32] Edgar Gabriel et al. “Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation”. In: *Proceedings, 11th European PVM/MPI Users’ Group Meeting*. Budapest, Hungary, 2004, pp. 97–104.
- [33] 10x genomics. *10x genomics site*. <https://www.10xgenomics.com/>. [Online; accessed 10-April-2025]. 2022.
- [34] Michael Hagemann-Jensen et al. “Single-cell RNA counting at allele and isoform resolution using Smart-seq3”. In: *Nature Biotechnology* 38.6 (2020), pp. 708–714.
- [35] Mahdi Hannoun et al. “MOISE: An organizational model for multi-agent systems”. In: *Ibero-American Conference on Artificial Intelligence*. Springer. 2000, pp. 156–165.
- [36] Stephanie C Hicks et al. “Missing data and technical variability in single-cell RNA-sequencing experiments”. In: *Biostatistics* 19.4 (2018), pp. 562–578.
- [37] Mingye Hong et al. “RNA sequencing: new technologies and applications in cancer research”. In: *Journal of hematology & oncology* 13.1 (2020), pp. 1–16.
- [38] Mingye Hong et al. “RNA sequencing: new technologies and applications in cancer research”. In: *Journal of hematology & oncology* 13.1 (2020), pp. 1–16.
- [39] Illumina. *Sequencing platform illumina MiSeq*. <https://emea.illumina.com/systems/sequencing-platforms/miseq.html>. [Online; accessed 6-April-2025]. 2022.

- [40] Saiful Islam et al. “Quantitative single-cell RNA-seq with unique molecular identifiers”. In: *Nature methods* 11.2 (2014), pp. 163–166.
- [41] Jon ET Jakobsson, Ola Spjuth, and Malin C Lagerström. “scConnect: a method for exploratory analysis of cell–cell communication based on single-cell RNA-sequencing data”. In: *Bioinformatics* 37.20 (2021), pp. 3501–3508.
- [42] Brandon Jew et al. “Accurate estimation of cell composition in bulk expression through robust integration of single-cell information”. In: *Nature communications* 11.1 (2020), pp. 1–11.
- [43] Suoqin Jin et al. “Inference and analysis of cell-cell communication using CellChat”. In: *Nature communications* 12.1 (2021), p. 1088.
- [44] Teemu Kivioja et al. “Counting absolute numbers of molecules using unique molecular identifiers”. In: *Nature methods* 9.1 (2012), pp. 72–74.
- [45] cole trapnell lab. *monocle 3 repository and documentation*. <https://cole-trapnell-lab.github.io/monocle3/>. [Online; accessed 10-April-2025]. 2022.
- [46] Grégory PA Lacraz et al. “Tomo-Seq identifies SOX9 as a key regulator of cardiac fibrosis during ischemic injury”. In: *Circulation* 136.15 (2017), pp. 1396–1409.
- [47] Lexogen. *Lexogen RNA selection*. <https://www.lexogen.com/polya-rna-selection-kit/>. [Online; accessed 6-April-2025]. 2022.
- [48] Yang Liao, Gordon K Smyth, and Wei Shi. “featureCounts: an efficient general purpose program for assigning sequence reads to genomic features”. In: *Bioinformatics* 30.7 (2014), pp. 923–930.
- [49] Giorgio Locicero et al. “MASFENON, a novel simulation framework for dynamic interconnected networks: a case study on epidemics networks”. In: *Available at SSRN 5472734* (2025).
- [50] Giorgio Locicero et al. “MASFENON: implementing a multi-agent simulation framework for interconnected networks with distributed programming”. In: *Procedia Computer Science* 255 (2025), pp. 73–82.
- [51] Giorgio Locicero et al. “MASFENON: Multi-Agent Adaptive Simulation Framework for Evolution in Networks of Networks”. In: *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE. 2023, pp. 3714–3720.
- [52] Romain Lopez et al. “Deep generative modeling for single-cell transcriptomics”. In: *Nature methods* 15.12 (2018), pp. 1053–1058.
- [53] Michael I Love, Wolfgang Huber, and Simon Anders. “Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2”. In: *Genome biology* 15.12 (2014), pp. 1–21.

- [54] David C Luckham and Brian Frasca. “Complex event processing in distributed systems”. In: *Computer Systems Laboratory Technical Report CSL-TR-98-754. Stanford University, Stanford* 28 (1998), p. 16.
- [55] Sean Luke et al. “Mason: A multiagent simulation environment”. In: *Simulation* 81.7 (2005), pp. 517–527.
- [56] Sai Ma et al. “Chromatin potential identified by shared single-cell profiling of RNA and chromatin”. In: *Cell* 183.4 (2020), pp. 1103–1116.
- [57] Charles M Macal and Michael J North. “Tutorial on agent-based modeling and simulation”. In: *Proceedings of the Winter Simulation Conference, 2005. IEEE.* 2005, 14–pp.
- [58] Salem Malikic et al. “Integrative inference of subclonal tumour evolution from single-cell and bulk sequencing data”. In: *Nature communications* 10.1 (2019), pp. 1–12.
- [59] S Manfredo Vieira et al. “Translocation of a gut pathobiont drives autoimmunity in mice and humans”. In: *Science* 359.6380 (2018), pp. 1156–1161.
- [60] Lucile Massenet-Regad and Vassili Soumelis. “ICELLNET v2: a versatile method for cell–cell communication analysis from human transcriptomic data”. In: *Bioinformatics* 40.3 (2024), btae089.
- [61] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).
- [62] Mark EJ Newman. “Modularity and community structure in networks”. In: *Proceedings of the national academy of sciences* 103.23 (2006), pp. 8577–8582.
- [63] Susanne Nichterwitz et al. “LCM-Seq: a method for spatial transcriptomic profiling using laser capture microdissection coupled with PolyA-based RNA sequencing”. In: *RNA Detection: Methods and Protocols* (2018), pp. 95–110.
- [64] Floriane Noël et al. “Dissection of intercellular communication using the transcriptome-based framework ICELLNET”. In: *Nature communications* 12.1 (2021), p. 1089.
- [65] Paul D O’Brien and Richard C Nicol. “FIPA—towards a standard for software agents”. In: *BT Technology Journal* 16.3 (1998), pp. 51–59.
- [66] Christopher A Odhams, Deborah S Cunninghame Graham, and Timothy J Vyse. “Profiling RNA-Seq at multiple resolutions markedly increases the number of causal eQTLs in autoimmune disease”. In: *PLoS genetics* 13.10 (2017), e1007071.

- [67] Jeffrey D Orth, Ines Thiele, and Bernhard Ø Palsson. “What is flux balance analysis?” In: *Nature biotechnology* 28.3 (2010), pp. 245–248.
- [68] Fatih Ozsolak and Patrice M Milos. “RNA sequencing: advances, challenges and opportunities”. In: *Nature reviews genetics* 12.2 (2011), pp. 87–98.
- [69] Michael P Papazoglou et al. “Service-oriented computing: State of the art and research challenges”. In: *Computer* 40.11 (2007), pp. 38–45.
- [70] Rob Patro et al. “Salmon provides fast and bias-aware quantification of transcript expression”. In: *Nature methods* 14.4 (2017), pp. 417–419.
- [71] Tao Peng et al. “SCRABBLE: single-cell RNA-seq imputation constrained by bulk RNA-seq data”. In: *Genome biology* 20.1 (2019), pp. 1–12.
- [72] Simone Picelli et al. “Full-length RNA-seq from single cells using Smart-seq2”. In: *Nature protocols* 9.1 (2014), pp. 171–181.
- [73] Qiagen. *Qiagen RNA selection*. <https://www.qiagen.com/kr/qec/psg-rna-selection-tools/>. [Online; accessed 6-April-2025]. 2022.
- [74] Steven F Railsback and Volker Grimm. *Agent-based and individual-based modeling: a practical introduction*. Princeton university press, 2019.
- [75] Anand S Rao. “AgentSpeak (L): BDI agents speak out in a logical computable language”. In: *European workshop on modelling autonomous agents in a multi-agent world*. Springer. 1996, pp. 42–55.
- [76] Jiahua Rao et al. “Imputing single-cell RNA-seq data by combining graph convolution and autoencoder neural networks”. In: *Iscience* 24.5 (2021), p. 102393.
- [77] Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. “edgeR: a Bioconductor package for differential expression analysis of digital gene expression data”. In: *bioinformatics* 26.1 (2010), pp. 139–140.
- [78] Alexander B Rosenberg et al. “Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding”. In: *Science* 360.6385 (2018), pp. 176–182.
- [79] Giulio Rossetti, Letizia Milli, and Salvatore Rinzivillo. “NDlib: a python library to model and analyze diffusion processes over complex networks”. In: *Companion Proceedings of the The Web Conference 2018*. 2018, pp. 183–186.
- [80] Conrad Sanderson and Ryan Curtin. “Armadillo: An efficient framework for numerical linear algebra”. In: *2025 17th International Conference on Computer and Automation Engineering (ICCAE)*. IEEE. 2025, pp. 303–307.

- [81] satijalab. *SCtransform vignette*. https://satijalab.org/seurat/articles/sctransform_vignette.html. [Online; accessed 29-April-2025]. 2025.
- [82] satijalab. *Seurat repository and documentation*. <https://satijalab.org/seurat/>. [Online; accessed 10-April-2025]. 2022.
- [83] Amanda N Scholes and Jeffrey A Lewis. “Comparison of RNA isolation methods on RNA-Seq: implications for differential expression and meta-analyses”. In: *BMC genomics* 21.1 (2020), pp. 1–9.
- [84] SingleR. *SingleR documentation*. <https://bioconductor.org/books/release/SingleRBook/>. [Online; accessed 10-April-2025]. 2022.
- [85] Rory Stark, Marta Grzelak, and James Hadfield. “RNA sequencing: the teenage years”. In: *Nature Reviews Genetics* 20.11 (2019), pp. 631–656.
- [86] Marlon Stoeckius et al. “Simultaneous epitope and transcriptome measurement in single cells”. In: *Nature methods* 14.9 (2017), pp. 865–868.
- [87] Peter Stone and Manuela Veloso. “Multiagent systems: A survey from a machine learning perspective”. In: *Autonomous Robots* 8 (2000), pp. 345–383.
- [88] Forrest Stonedahl and Uri Wilensky. “NetLogo Virus on a Network model”. In: *Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL* (2008).
- [89] Kelly Street et al. “Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics”. In: *BMC genomics* 19.1 (2018), p. 477.
- [90] Patrick Taillandier et al. “Building, composing and experimenting complex spatial models with the GAMA platform”. In: *GeoInformatica* 23.2 (2019), pp. 299–322.
- [91] Fuchou Tang et al. “mRNA-Seq whole-transcriptome analysis of a single cell”. In: *Nature methods* 6.5 (2009), pp. 377–382.
- [92] Seth Tisue and Uri Wilensky. “Netlogo: A simple environment for modeling complexity”. In: *International conference on complex systems*. Vol. 21. Citeseer. 2004, pp. 16–21.
- [93] Kevin Troulé et al. “CellPhoneDB v5: inferring cell–cell communication from single-cell multiomics data”. In: *Nature Protocols* (2025), pp. 1–29.
- [94] Jialin Wang and Zhihua Duan. “Agent ai with langgraph: A modular framework for enhancing machine translation using large language models”. In: *arXiv preprint arXiv:2412.03801* (2024).

- [95] Jinglu Wang et al. “RNA sequencing (RNA-Seq) and its application in ovarian cancer”. In: *Gynecologic oncology* 152.1 (2019), pp. 194–201.
- [96] Duncan J Watts and Steven H Strogatz. “Collective dynamics of ‘small-world’ networks”. In: *nature* 393.6684 (1998), pp. 440–442.
- [97] Wikipedia contributors. *Gene regulatory network* — *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Gene_regulatory_network&oldid=1106222025. [Online; accessed 26-April-2025]. 2022.
- [98] Jane AC Wilson et al. “RNA-Seq analysis of chikungunya virus infection and identification of granzyme A as a major promoter of arthritic inflammation”. In: *PLoS pathogens* 13.2 (2017), e1006155.
- [99] F Alexander Wolf et al. “PAGA: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells”. In: *Genome biology* 20.1 (2019), p. 59.
- [100] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & sons, 2009.
- [101] Qingyun Wu et al. “Autogen: Enabling next-gen LLM applications via multi-agent conversations”. In: *First Conference on Language Modeling*. 2024.
- [102] Benjamin A Yang et al. “Engineered tools to study intercellular communication”. In: *Advanced science* 8.3 (2021), p. 2002825.
- [103] Yingxuan Yang et al. “Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems”. In: *arXiv preprint arXiv:2504.00587* (2025).
- [104] Junwei Yu, Yepeng Ding, and Hiroyuki Sato. “DynTaskMAS: A Dynamic Task Graph-driven Framework for Asynchronous and Parallel LLM-based Multi-Agent Systems”. In: *arXiv preprint arXiv:2503.07675* (2025).
- [105] Naouel Zerrouk, Franck Augé, and Anna Niarakis. “Building a modular and multi-cellular virtual twin of the synovial joint in Rheumatoid Arthritis”. In: *npj Digital Medicine* 7.1 (2024), p. 379.
- [106] Yingdong Zhao et al. “TPM, FPKM, or normalized counts? A comparative study of quantification measures for the analysis of RNA-seq data from the NCI patient-derived models repository”. In: *Journal of translational medicine* 19.1 (2021), pp. 1–15.