Neurodynamic chance-constrained geometric optimization

Siham Tassouli

université PARIS-SACLAY

Neurodynamic Chance-Constrained Geometric Optimization

Optimisation Géométrique Neurodynamique avec des Contraintes en Probabilité

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 : Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat: Informatique mathématique
Graduate School : Informatique et Sciences du Numérique
Référent : Faculté des sciences d'Orsay

Thèse préparée dans le **Laboratoire des signaux et systèmes** (Université Paris-Saclay, CNRS, CentraleSupélec), sous la direction d'**Abdel LISSER**, Professeur, Université Paris-Saclay

Thèse soutenue à Paris-Saclay, le 12 Octobre 2023, par

Siham TASSOULI

Composition du jury

Membres du jury avec voix délibérative

Antoine GIRARD Directeur de recherche, CentraleSupelec, France	Président
Janny LEUNG Professeure, University of Macau, Chine	Rapporteure & Examinatrice
Hasnaa ZIDANI Professeure, INSA Rouen, France	Rapporteure & Examinatrice
Rossana RICCARDI Professeure associée, University of Brescia, Italie	Examinatrice

THESE DE DOCTORAT

NNT : 2023UPASG062

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Optimisation Géométrique Neurodynamique avec des Contraintes en Probabilité
Mots clés: Optimisation géométrique, Contraintes en probabilité, Réseaux de neurones récurrents, Théorie de Lyapunov, Equations différentielles ordinaires.

Résumé: Dans de nombreux problèmes réels, les décideurs sont confrontés à des incertitudes qui peuvent affecter les résultats de leurs décisions. Ces incertitudes découlent de diverses sources, telles que la variabilité de la demande, les conditions fluctuantes du marché ou des informations incomplètes sur les paramètres du système. Les approches traditionnelles d'optimisation déterministe supposent que tous les paramètres sont connus avec certitude, ce qui peut ne pas refléter avec précision la réalité du problème. L'optimisation sous contraintes de probabilité offre une approche plus réaliste et robuste en tenant compte explicitement de l'incertitude dans la prise de décision. La programmation géométrique est souvent mal comprise comme une technique exclusivement conçue pour les problèmes posynômes. Cependant, c'est une théorie mathématique polyvalente qui a une valeur significative pour résoudre un large éventail de problèmes. En fait, sa véritable force réside dans sa capacité à résoudre efficacement des problèmes en apparence inséparables en exploitant leur structure algébrique linéaire. Cette applicabilité générale de la programmation géométrique en fait un outil précieux pour étudier et résoudre divers problèmes d'optimisation, étendant ainsi son utilité pratique au-delà de sa perception initiale. Les réseaux de neurones récurrents offrent un cadre de calcul inspiré de la biologie avec un grand potentiel d'optimisation. En imitant la structure interconnectée des neurones du cerveau, les réseaux de neurones récurrents excellent dans la modélisation de systèmes complexes et dynamiques. Cette capacité leur permet de capturer les dépendances temporelles et les boucles de rétroaction, ce qui les rend bien adaptés aux scénarios d'optimisation impliquant des prises de décision séquentielles ou des processus itératifs. De plus, l'un des principaux avantages des approches neurodynamiques est leur faisabilité de mise en œuvre matérielle. L'objectif principal de cette thèse est de développer des algorithmes neurodynamiques efficaces et performants pour résoudre des problèmes d'optimisation géométrique avec des contraintes de probabilité. La thèse commence par les programmes géométriques avec des contraintes de probabilité impliquant des variables aléatoires indépendantes. De plus, un type spécifique de programmes géométriques appelés programmes rectangulaires est également examiné en détail. L'objectif est de comprendre les caractéristiques et les complexités associées à cette sous-classe de programmes géométriques. Ensuite, la thèse explore l'application de la théorie des copules pour aborder les programmes géométriques avec des contraintes de probabilité impliquant des variables aléatoires dépendantes. La théorie des copules fournit un cadre mathématique pour modéliser et analyser la structure de dépendance entre les variables aléatoires, améliorant ainsi la compréhension et l'optimisation de ces problèmes. Enfin, la thèse examine l'optimisation géométrique robuste, qui prend en compte les distributions incertaines des variables aléatoires. Cette approche vise à développer des algorithmes d'optimisation résistant à l'incertitude dans les distributions de probabilité sous-jacentes, garantissant des solutions plus fiables et stables.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Neurodynamic Chance-Constrained Geometric Optimization
Keywords: Geometric optimization, Chance constraints, Recurrent neural networks, Lyapunov Theory, Ordinary differential equations

Abstract: In many real-world scenarios, decision-makers face uncertainties that can affect the outcomes of their decisions. These uncertainties arise from various sources, such as variability in demand, fluctuating market conditions, or incomplete information about system parameters. Traditional deterministic optimization approaches assume that all parameters are known with certainty, which may not accurately reflect the reality of the problem. Chance-constrained optimization provides a more realistic and robust approach by explicitly accounting for the uncertainty in decision-making. Geometric programming is often misunderstood as a technique exclusively designed for posynomial problems. However, it is a versatile mathematical theory with significant value in addressing a broad range of separable problems. In fact, its true strength lies in its ability to effectively tackle seemingly inseparable problems by leveraging their linear algebraic structure. This general applicability of geometric programming makes it a valuable tool for studying and solving various optimization problems, extending its practical usefulness beyond its initial perception. Recurrent neural networks (RNNs) offer a biologically inspired computational framework with great optimization potential. By emulating the interconnected structure of neurons in the brain, RNNs excel in modeling complex and dynamic systems. This capability allows them to capture temporal dependencies and feedback loops, making them well-suited for optimization scenarios that involve sequential decision-making or iterative processes. Moreover, one of the key advantages of neurodynamic approaches is their hardware implementation feasibility. The primary objective of this thesis is to develop neurodynamic algorithms that are efficient and effective in solving chance-constrained geometric optimization problems. The thesis begins by focusing on chance-constrained geometric programs involving independent random variables. In addition, a specific type of geometric programs known as rectangular programs is also examined in detail. The objective is to understand the characteristics and complexities associated with this subclass of geometric programs. Subsequently, the thesis explores applying copula theory to address chance-constrained geometric programs with dependent random variables. Copula theory provides a mathematical framework for modeling and analyzing the dependence structure between random variables, thereby enhancing the understanding and optimization of these problems. Lastly, the thesis investigates distributionally robust geometric optimization, which considers uncertain distributions of random variables. This approach focuses on developing optimization algorithms that are robust against uncertainty in the underlying probability distributions, ensuring more reliable and stable solutions.

Synthèse

Dans son exploration des techniques visant à minimiser les coûts dans les problèmes de conception en ingénierie, Zener a initié les premières recherches dans ce qui est maintenant connu sous le nom de programmation géométrique. Le travail de Zener, ainsi que les articles ultérieurs de Duffin et Peterson ont posé les bases fondamentales de ce domaine. Le terme "programmation géométrique" a été adopté en raison du rôle significatif joué par l'inégalité arithmético-géométrique dans son développement initial. Au départ, la programmation géométrique se concentrait principalement sur la minimisation de fonctions posynomiales tout en respectant des contraintes d'inégalité. Ainsi, on aurait pu l'appeler programmation posynomiale plutôt que programmation géométrique. La programmation géométrique offre un moyen d'exprimer et d'analyser de nombreux problèmes d'optimisation importants dans un format séparable, même lorsqu'ils sont généralement considérés comme inséparables. L'aspect fondamental de cette approche réside dans l'exploitation des propriétés linéaires inhérentes au problème en cours.

Traditionnellement, la programmation géométrique a été appliquée à des problèmes avec des valeurs de paramètres connues et précises. Cependant, dans la vie réelle, les valeurs observées de ces paramètres sont souvent imprécises. Ces données incertaines peuvent prendre différentes formes, notamment des plages bornées, des intervalles, des ensembles flous ou des variables aléatoires. Afin de gérer ces incertitudes, des extensions et des modifications de l'approche traditionnelle de la programmation géométrique ont été développées. Ces variantes intègrent des techniques de l'analyse par intervalles, de la programmation stochastique ou de l'optimisation robuste. En tenant compte de la nature incertaine des paramètres, ces approches fournissent des solutions plus robustes et fiables aux problèmes de programmation géométrique de la vie réelle. Par conséquent, avec l'incorporation de données incertaines ou imprécises, la programmation géométrique devient un outil polyvalent et adaptable pour résoudre des problèmes d'optimisation dans des applications pratiques.

Dans la programmation géométrique stochastique, il est admis que certains coefficients et/ou exposants dans le problème de programmation géométrique ne sont pas connus précisément, et leur connaissance incomplète est représentée à l'aide de modèles probabilistes ou aléatoires. L'incertitude associée à ces coefficients peut provenir de diverses sources telles que des erreurs de mesure, de la variabilité des données ou du caractère intrinsèquement aléatoire du système modélisé. Au lieu de supposer des valeurs déterministes pour ces paramètres incertains, la programmation géométrique stochastique intègre leur caractère aléatoire dans la formulation du problème.

Tout au long de la vie, les réseaux cérébraux sont constamment en effervescence avec des ondes d'activité. D'innombrables signaux s'harmonisent et oscillent, formant l'architecture fonctionnelle sous-jacente qui façonne chaque aspect de l'existence. Les émotions, les niveaux de stress, les préférences musicales, les aspirations et les rêves trouvent leur origine dans le comportement dynamique de ces réseaux, complété par

des systèmes de mémoire adaptative qui cherchent à mieux s'aligner sur l'environnement en constante évolution de chacun. L'ampleur considérable et l'intégration complexe des réseaux dans le cerveau humain posent des défis redoutables pour comprendre si des formes traditionnelles de calcul s'y produisent et comment. Les preuves indiquent que le traitement de l'information se déroule à divers niveaux, englobant des protéines modulatrices au sein des cellules individuelles, des microcircuits corticaux et des réseaux fonctionnels étendus couvrant l'ensemble du cerveau. Cependant, la compréhension expérimentale du fonctionnement du cerveau progresse lentement. Heureusement, des ingénieurs ingénieux ont élaboré des algorithmes qui simulent partiellement certains aspects de ces réseaux. Bien qu'aucun modèle unique ne puisse pleinement encapsuler la complexité pure et le comportement du cerveau humain, ces outils offrent aux chercheurs une précieuse fenêtre sur la manière dont l'information pourrait être calculée et finalement représentée dans l'activité de réseaux distribués.

Une catégorie importante de problèmes logiques découlant de scénarios du monde réel peut être formulée comme des problèmes d'optimisation, qui peuvent être compris qualitativement comme une quête de la solution optimale. Ces problèmes sont courants dans des domaines tels que l'ingénierie et le commerce, ainsi que dans des défis perceptuels qui nécessitent une résolution rapide par les systèmes nerveux. Le concept fondamental derrière l'approche des réseaux neuronaux pour l'optimisation est de créer une fonction d'énergie non négative et d'établir un système dynamique qui imite un réseau neuronal artificiel. Ce système dynamique prend généralement la forme d'équations différentielles ordinaires du premier ordre. L'objectif est que le système dynamique converge vers un état statique ou un point d'équilibre, qui correspond à la solution du problème d'optimisation sous-jacent, à partir d'un point initial. Un avantage notable de l'utilisation de réseaux neuronaux pour résoudre des problèmes d'optimisation est leur implémentabilité matérielle. En d'autres termes, ces réseaux neuronaux peuvent être mis en œuvre à l'aide de circuits intégrés. De plus, les réseaux neuronaux conçus pour une mise en œuvre par circuit offrent des capacités de traitement en temps réel, renforçant ainsi leur applicabilité pratique.

Cette thèse vise à explorer l'application des approches neurodynamiques dans le domaine de l'optimisation géométrique sous contraintes de probabilité. La thèse se concentre sur le développement d'algorithmes efficaces et de méthodes de solution pour résoudre des problèmes d'optimisation complexes avec des contraintes géométriques dans des conditions incertaines. La thèse se compose de sept chapitres, comprenant une revue de littérature complète, un contexte théorique et diverses études de cas.

Le chapitre 2 propose une revue de littérature sur l'optimisation géométrique. Le chapitre discute des approches de solution et des algorithmes pour résoudre des programmes géométriques, y compris les méthodes duales et primales. Divers algorithmes et techniques ont été passés en revue, tels que la condensation, les méthodes à points intérieurs et les algorithmes primaux-duaux. Le chapitre explore également l'application de la programmation géométrique dans différents domaines, notamment l'ingénierie mécanique, l'ingénierie chimique, le contrôle de puissance et la finance. De plus, des exten-

sions de la programmation géométrique sont discutées, qui assouplissent la contrainte posynomiale et élargissent la gamme de fonctions pouvant être optimisées. Le chapitre se conclut par une introduction à l'optimisation sous contraintes de probabilité, également connue sous le nom de programmation probabiliste ou stochastique, qui traite des incertitudes et des risques dans la prise de décision. Il donne un aperçu des concepts de programmation sous contraintes de probabilité, des résultats établis et des approches computationnelles pour résoudre les problèmes d'optimisation sous contraintes de probabilité. Les applications du monde réel de la programmation sous contraintes de probabilité sont également examinées, mettant en évidence son utilité dans la gestion des incertitudes et la prise de décisions robustes.

Le chapitre 3 fournit un contexte théorique sur les réseaux neuronaux dynamiques et l'optimisation biconvexe. Le chapitre commence par mettre en avant la complexité et l'adaptabilité des réseaux cérébraux, ce qui inspire l'utilisation de réseaux neuronaux pour l'optimisation. Il met en évidence le potentiel des réseaux neuronaux pour imiter des systèmes dynamiques et converger vers des solutions optimales. Le chapitre explore l'histoire des modèles de réseaux neuronaux pour l'optimisation, en commençant par le modèle de Hopfield et Tank pour le Problème du Voyageur de Commerce. Il discute des extensions et des avancées réalisées dans les approches de réseaux neuronaux, notamment l'incorporation de paramètres de pénalité et le développement de modèles pour les problèmes d'optimisation contraints et non lisses. Plusieurs modèles de réseaux neuronaux spécifiques sont présentés, chacun conçu pour aborder différents types de problèmes d'optimisation. Ces modèles comprennent des réseaux neuronaux récurrents basés sur des pénalités pour les problèmes d'optimisation contraints, des approches neurodynamiques pour les problèmes d'optimisation convexe avec des contraintes générales, et des réseaux récurrents à une couche pour les problèmes d'optimisation convexe non lisses. Les avantages de ces modèles de réseaux neuronaux sont mis en avant, tels que leur efficacité computationnelle, leur simplicité, et leur capacité à traiter une large gamme de problèmes d'optimisation. Le chapitre présente également les applications des réseaux neuronaux récurrents dans des problèmes du monde réel, tels que l'optimisation de portefeuille dynamique, l'allocation de force de freinage électro-hydraulique, le pronostic de la santé des éoliennes, et le fonctionnement optimal des microgrids électriques. Pour offrir une compréhension plus approfondie des réseaux neuronaux récurrents, deux modèles spécifiques sont présentés dans les sous-sections suivantes. Ces modèles démontrent l'application des réseaux neuronaux récurrents dans la résolution de problèmes d'optimisation non linéaires convexes. Le chapitre passe ensuite à l'optimisation biconvexe. Nous mettons en évidence plusieurs avancées et applications de l'optimisation biconvexe au cours des dernières années. Le chapitre souligne la nature complexe de la résolution des problèmes de programmation biconvexe, nécessitant l'exploration de nouvelles théories et méthodes de solution. Dans cette section, deux résultats théoriques significatifs liés à l'optimum partiel de la programmation biconvexe sont présentés, en utilisant la fonction de pénalité de l'objectif. Le premier résultat établit l'équivalence entre la condition partielle de Karush-Kuhn-Tucker (KKT) et la propriété d'exactitude par-

tielle de la fonction de pénalité de l'objectif dans la programmation biconvexe. Cette découverte fournit un aperçu précieux de la relation entre la condition KKT et le comportement de la fonction de pénalité de l'objectif, permettant une compréhension plus approfondie du processus d'optimisation. Le deuxième résultat démontre l'équivalence entre la condition de stabilité partielle et la propriété d'exactitude partielle de la fonction de pénalité de l'objectif dans la programmation biconvexe. Ce résultat met en lumière l'importance de la condition de stabilité par rapport à la fonction de pénalité de l'objectif, offrant des informations précieuses sur le comportement de convergence des algorithmes visant à résoudre l'optimum partiel de la programmation biconvexe. Ces résultats théoriques offrent des garanties cruciales pour la convergence des algorithmes conçus pour traiter l'optimum partiel des problèmes de programmation biconvexe. En établissant l'équivalence entre les conditions clés et les propriétés de la fonction de pénalité de l'objectif, cette thèse fournit une base solide pour le développement d'algorithmes efficaces et performants capables de converger de manière fiable vers des solutions optimales pour la programmation biconvexe.

Le chapitre 4 se concentre sur les problèmes d'optimisation géométrique conjoints sous contraintes de probabilité où les coefficients suivent une distribution normale et les vecteurs de lignes de la matrice sont indépendants. Le chapitre commence par introduire les programmes géométriques et leurs formulations équivalentes déterministes. Ensuite, le chapitre explore le concept d'approximations convexes, qui sont des techniques utilisées pour transformer des problèmes d'optimisation non convexes en problèmes convexes. En approximant le problème d'origine par une formulation convexe, il devient possible d'appliquer des algorithmes d'optimisation convexes existants pour trouver des solutions optimales. Cette approche améliore la traitabilité des problèmes d'optimisation géométrique conjoints sous contraintes de probabilité. Contrairement aux méthodes existantes qui reposent sur des approximations convexes, nous proposons une approche de réseau neuronal dynamique récurrent pour résoudre le programme géométrique stochastique. L'efficacité de cette approche est évaluée à travers des expériences numériques, notamment un problème d'optimisation de forme tridimensionnelle et un problème de transport multidimensionnel. Ces expériences démontrent la capacité de l'approche du réseau neuronal dynamique à traiter efficacement les problèmes d'optimisation géométrique conjoints sous contraintes de probabilité. De plus, le chapitre introduit une approche neurodynamique spécialement conçue pour résoudre des programmes rectangulaires, qui sont un cas particulier des programmes géométriques. Nous transformons le problème stochastique en un problème déterministe et utilisons une transformation logarithmique combinée avec l'inégalité arithmético-géométrique pour le convertir en un problème biconvexe. L'approche proposée utilise des réseaux neuronaux pour mettre à jour de manière itérative les variables du problème d'optimisation et trouver des solutions optimales. En appliquant l'approche neurodynamique, le chapitre démontre la capacité à résoudre efficacement des programmes rectangulaires. Enfin, le chapitre présente une étude de cas sur la maximisation du Rapport Signal sur Bruit d'Interférence (SINR) pour les systèmes Massive Multiple Input Multiple Output (MIMO).

Cette étude de cas met en évidence l'application pratique de l'optimisation géométrique conjointe sous contraintes de probabilité dans les systèmes de communication sans fil. L'objectif est d'optimiser l'allocation de ressources dans un système MIMO pour maximiser le SINR, améliorant ainsi les performances du système.

Le chapitre 5 se concentre sur les problèmes d'optimisation conjointe sous contraintes de probabilité dépendantes. Il introduit la théorie des copules et les vecteurs aléatoires elliptiquement symétriques comme fondements théoriques pour la modélisation et la résolution de ces problèmes. Le chapitre présente une approche de réseau neuronal dynamique pour résoudre les programmes linéaires et les programmes géométriques, en fournissant des formulations équivalentes déterministes pour les deux. Des expériences numériques sont menées pour valider l'efficacité et l'efficience de l'approche proposée dans la résolution de problèmes d'optimisation conjointe sous contraintes de probabilité dépendantes dans des applications pratiques. Le chapitre commence par introduire la théorie des copules et les vecteurs aléatoires elliptiquement symétriques comme base théorique pour aborder de tels problèmes d'optimisation. Ces concepts fournissent la fondation pour la modélisation et la résolution de problèmes impliquant des variables aléatoires dépendantes. Le chapitre explore ensuite les programmes linéaires et leurs formulations équivalentes déterministes dans le contexte des problèmes d'optimisation conjointe sous contraintes de probabilité dépendantes. En formulant le problème de manière déterministe, il devient plus adaptable aux techniques de résolution. Nous proposons ensuite une approche de réseau neuronal dynamique pour résoudre efficacement ces programmes linéaires. Cette approche offre une méthode novatrice pour traiter les problèmes d'optimisation conjointe sous contraintes de probabilité dépendantes sans recourir aux algorithmes d'optimisation conventionnels. Pour valider l'efficacité de la méthode proposée, le chapitre présente des expériences numériques. Ces expériences impliquent la résolution de différents problèmes d'optimisation conjointe sous contraintes de probabilité dépendantes en utilisant l'approche de réseau neuronal dynamique proposée. En plus des programmes linéaires, le chapitre couvre également les programmes géométriques. Le chapitre présente les formulations équivalentes déterministes pour les programmes géométriques et étend l'approche du réseau neuronal dynamique pour résoudre ces problèmes. Cette extension permet la résolution efficace de problèmes d'optimisation géométrique conjointe sous contraintes de probabilité dépendantes. Tout comme dans le chapitre précédent, des expériences numériques sont menées pour évaluer la performance de la méthode proposée pour les programmes géométriques. Ces expériences visent à évaluer l'exactitude et l'efficacité de la méthode dans la résolution de problèmes d'optimisation géométrique conjointe sous contraintes de probabilité dépendantes dans des scénarios pratiques.

Le chapitre 6 examine les problèmes d'optimisation géométrique robustes à la distribution. Il introduit le sujet et se concentre sur les programmes linéaires, en proposant une approche neurodynamique duplex pour résoudre ces problèmes. Le chapitre fournit une analyse de la convergence et mène des expériences numériques pour évaluer les performances de la méthode proposée. De plus, le chapitre explore les programmes

géométriques robustes à la distribution avec des ensembles d'incertitude définis par les deux premiers moments d'ordre et présente une approche dynamique de réseau neuronal récurrent pour les résoudre. Des expériences numériques sont menées pour valider l'efficacité de cette approche. Le chapitre commence par une introduction au sujet, en donnant un aperçu des défis et des motivations derrière l'optimisation robuste à la distribution. Le chapitre se concentre ensuite sur les programmes linéaires et présente une approche neurodynamique duplex pour résoudre les problèmes d'optimisation géométrique robustes à la distribution dans ce contexte. L'approche proposée utilise des architectures de réseaux neuronaux et des systèmes dynamiques pour mettre à jour de manière itérative les variables de décision et atteindre la convergence. Le chapitre fournit une analyse détaillée de la convergence, démontrant la stabilité et les propriétés de convergence de l'approche neurodynamique duplex. Pour évaluer les performances de la méthode proposée, le chapitre mène des expériences numériques. Ces expériences résolvent divers programmes linéaires robustes à la distribution en utilisant l'approche neurodynamique duplex. Les résultats des expériences sont analysés pour évaluer l'exactitude, l'efficacité et l'efficacité de la méthode proposée dans des scénarios pratiques. En plus des programmes linéaires, le chapitre explore les programmes géométriques robustes à la distribution avec des ensembles d'incertitude définis par les deux premiers moments d'ordre. Le chapitre présente une approche dynamique de réseau neuronal récurrent spécialement conçue pour résoudre ces programmes géométriques robustes à la distribution. Cette approche tire parti de l'architecture de réseau neuronal récurrent pour capturer la dynamique temporelle du processus d'optimisation. Des expériences numériques sont menées pour valider l'efficacité de l'approche proposée pour les programmes géométriques robustes à la distribution. Ces expériences résolvent divers problèmes d'optimisation avec des ensembles d'incertitude définis par les deux premiers moments d'ordre en utilisant l'approche dynamique de réseau neuronal récurrent. Les résultats des expériences fournissent des informations sur les performances et l'efficacité de la méthode proposée pour résoudre les problèmes d'optimisation géométrique robustes à la distribution.

Le chapitre 7 sert de résumé complet des principales découvertes et contributions présentées tout au long de la thèse. Il commence par revisiter les objectifs de recherche et met en évidence l'importance des approches neurodynamiques proposées pour résoudre les problèmes d'optimisation géométrique sous contraintes de probabilité. Le chapitre résume ensuite les principales contributions de la recherche, en mettant l'accent sur les méthodologies et techniques novatrices développées pour résoudre différents types de problèmes d'optimisation géométrique. Il discute des avantages des approches neurodynamiques proposées, tels que leur capacité à gérer la non-convexité, à traiter des contraintes probabilistes conjointes et à fournir des solutions robustes. De plus, le chapitre explore les implications et les applications potentielles des approches neurodynamiques proposées. Il discute de la manière dont ces approches peuvent être appliquées dans des scénarios réels dans différents domaines, tels que le transport, l'optimisation de forme, le traitement du signal, et bien plus encore. Les avantages potentiels de l'intégration de ces approches dans les processus de prise de décision sont mis en évidence, no-

tamment l'amélioration de l'exactitude, de l'efficacité et de la robustesse des solutions d'optimisation. Dans la section de conclusion du chapitre, des orientations futures possibles pour la recherche dans le domaine de l'optimisation géométrique sont discutées. Ces orientations peuvent inclure une investigation plus approfondie de types spécifiques de programmes géométriques, l'exploration de l'intégration de contraintes ou objectifs supplémentaires, l'extension des approches neurodynamiques pour traiter des problèmes à plus grande échelle, ou l'adaptation des méthodologies à des domaines d'application spécifiques.

Acknowledgements

"Sometimes, others help us by simply being in our lives." - Dr. Ahmed Khaled Tawfik

At the forefront of my appreciation stands Professor Abdel Lisser, whose mentorship has exceeded the conventional bounds of a supervisor. His support, guidance, and wholehearted assistance across all facets of my research have left an indelible mark on completing this work. His mentorship is beyond measure, and I consider myself very fortunate to have been supervised during those three years by Professor Lisser.

I would also like to extend my thanks to the members of the jury, for their time, expertise, contributions and thoughtful feedback. I would also like to acknowledge my colleagues, Dawen, Shangyuan and Nam, whose insightful dialogues and thought-provoking discussions have been a constant source of inspiration. The collaborative atmosphere we shared has pushed me to engage in critical thinking and has added depth to my research expedition.

To my parents, Mom and Dad, I owe an immeasurable debt of gratitude for their unconditional love, support, and the countless sacrifices they have made on my behalf shaping not only my academic pursuits but the very essence of my being.

I extend my warmest thanks to my wonderful mother-in-law, whose tender morning messages and steadfast support have been a source of immeasurable comfort and positivity throughout my days. Her thoughtful gestures and continuous encouragement have woven a tapestry of warmth and brightness into my daily life.

I must take a moment to express my profound gratitude to my beloved sisters, Chaimae and Oumayma, for the presence they have maintained and the unflagging support they have provided throughout this remarkable endeavor. Their enduring companionship has been a beacon of strength, a steady source of encouragement, and a reminder of the profound bonds that family shares during both the joys and challenges of life's journey.

Lastly, I want to convey my deepest gratitude to my husband, Badreddine. His understanding, unconditional support and boundless love have been my steadfast pillars of strength. I am profoundly thankful for being my confidant, my cheerleader, my refuge in times of both celebration and uncertainty and simply for his enduring presence in my life.

To all those mentioned and to the countless others who have contributed to my academic and personal growth, your roles have been immeasurable. This thesis is a testament to the support and love that have surrounded me, and I am forever grateful for each of you.

with love, Siham

Contents

1 - Introduction

This thesis aims to explore the application of neurodynamic approaches in the field of chance-constrained geometric optimization. The thesis focuses on developing efficient algorithms and solution methods for solving complex optimization problems with geometric constraints under uncertain conditions. The thesis consists of seven chapters, including a comprehensive literature review, theoretical background, and various case studies. This introduction provides an overview of the research plan and the key elements discussed in each chapter.

Chapter 2 provides a literature review on geometric optimization. The chapter discusses solution approaches and algorithms for solving geometric programs, including dual and primal methods. Various algorithms and techniques have been reviewed, such as condensation, interior-point methods, and primal-dual algorithms. The chapter also explores the application of geometric programming in different domains, including mechanical engineering, chemical engineering, power control, and finance. Furthermore, extensions to geometric programming are discussed, which relax the posynomial constraint and expand the range of functions that can be optimized. The chapter concludes with an introduction to chance-constrained optimization, also known as probabilistic or stochastic programming, which addresses uncertainties and risks in decision-making. It provides an overview of chance-constrained programming concepts, established results, and computational approaches for solving chance-constrained optimization problems. Real-world applications of chance-constrained programming are also surveyed, highlighting its usefulness in handling uncertainties and making robust decisions.

Chapter 3 provides a theoretical background on dynamical neural networks and biconvex optimization. The chapter begins by emphasizing the complexity and adaptability of the brain's networks, which inspire the use of neural networks for optimization. It highlights the potential of neural networks to mimic dynamic systems and converge to optimal solutions. The chapter explores the history of neural network models for optimization, starting with Hopfield and Tank's model for the Traveling Salesman Problem. It discusses the extensions and advancements made in neural network approaches, including the incorporation of penalty parameters and the development of models for constrained and nonsmooth optimization problems. Several specific neural network models are presented, each designed to address different types of optimization problems. These models include penalty-based recurrent neural networks for constrained optimization problems, neurodynamic approaches for convex optimization problems with general constraints, and one-layer recurrent networks for non-smooth convex optimization problems. The advantages of these neural network models are highlighted, such as their computational efficiency, simplicity, and ability to handle a broad range of optimization problems. The chapter also showcases the applications of recurrent neural networks in real-world problems, such as dynamic portfolio optimization, electro-hydraulic braking force allocation, wind turbine health prognosis, and optimal operation of electrical microgrids. To provide

a deeper understanding of recurrent neural networks, two specific models are presented in the subsequent subsections. These models demonstrate the application of recurrent neural networks in solving convex nonlinear optimization problems. The chapter then transitions to biconvex optimization. We highlight several advancements and applications of biconvex optimization in recent years. The chapter emphasizes the challenging nature of solving biconvex programming problems, necessitating the exploration of new theories and solution methods. In this section, two significant theoretical results related to the partial optimum of biconvex programming are presented, utilizing the objective penalty function. The first result establishes the equivalence between the partial Karush-Kuhn-Tucker (KKT) condition and the partial exactness property of the objective penalty function in biconvex programming. This finding provides a valuable insight into the relationship between the KKT condition and the behavior of the objective penalty function, enabling a deeper understanding of the optimization process. The second result demonstrates the equivalence between the partial stability condition and the partial exactness property of the objective penalty function in biconvex programming. This result sheds light on the significance of the stability condition in relation to the objective penalty function, offering valuable insights into the convergence behavior of algorithms aiming to solve the partial optimum of biconvex programming. These theoretical results offer crucial guarantees for the convergence of algorithms designed to tackle the partial optimum of biconvex programming problems. By establishing the equivalence between key conditions and properties of the objective penalty function, this thesis provides a solid foundation for developing efficient and effective algorithms that can reliably converge towards optimal solutions for biconvex programming.

Chapter 4 focuses on joint chance-constrained geometric optimization problems where the coefficients are normally distributed, and the matrix row vectors are independent. The chapter starts by introducing geometric programs and their deterministic equivalent formulations. The chapter then explores the concept of convex approximations, which are techniques used to transform non-convex optimization problems into convex ones. By approximating the original problem with a convex formulation, it becomes possible to apply existing convex optimization algorithms to find optimal solutions. This approach enhances the tractability of joint chance-constrained geometric optimization problems. Unlike existing methods that rely on convex approximations, we propose a recurrent dynamical neural network approach to solve the stochastic geometric program. The effectiveness of this approach is evaluated through numerical experiments, including a three-dimensional shape optimization problem and a multidimensional transportation problem. These experiments demonstrate the ability of the dynamical neural network approach to handle joint chance-constrained geometric optimization problems efficiently. Furthermore, the chapter introduces a neurodynamic approach specifically designed to solve rectangular programs, which are a special case of geometric programs. We transform the stochastic problem into a deterministic one and utilize a logarithmic transformation combined with the arithmetic-geometric mean inequality to convert it into a biconvex problem. The proposed approach utilizes neural networks to iteratively update the vari-

ables of the optimization problem and find optimal solutions. By applying the neurodynamic approach, the chapter demonstrates the capability to solve rectangular programs effectively. Finally, the chapter presents a case study on maximizing the Signal to Interference Noise Ratio (SINR) for Massive Multiple Input Multiple Output (MIMO) systems. This case study highlights the practical application of joint chance-constrained geometric optimization in wireless communication systems. The goal is to optimize the allocation of resources in a MIMO system to maximize the SINR, thereby improving the system's performance.

Chapter 5 focuses on dependent joint chance-constrained optimization problems. It introduces copula theory and elliptically symmetric random vectors as the theoretical foundations for modeling and solving these problems. The chapter presents a dynamical neural network approach to solve linear programs and geometric programs, providing deterministic equivalent formulations for both. Numerical experiments are conducted to validate the effectiveness and efficiency of the proposed approach in solving dependent joint chance-constrained optimization problems in practical applications. The chapter begins by introducing copula theory and elliptically symmetric random vectors as the theoretical underpinnings for addressing such optimization problems. These concepts provide the foundation for modeling and solving problems involving dependent random variables. The chapter proceeds to explore linear programs and their deterministic equivalent formulations in the context of dependent joint chance-constrained optimization problems. By formulating the problem deterministically, it becomes more amenable to solution techniques. We then propose a dynamical neural network approach to solve these linear programs efficiently. This approach offers a novel method to handle dependent joint chance-constrained optimization problems without resorting to conventional optimization algorithms. To validate the effectiveness of the proposed method, the chapter presents numerical experiments. These experiments involve solving various dependent joint chance-constrained optimization problems using the proposed dynamical neural network approach. In addition to linear programs, the chapter also covers geometric programs. The chapter presents the deterministic equivalent formulations for geometric programs and extends the dynamical neural network approach to address these problems. This extension allows for the efficient solution of dependent joint chance-constrained geometric optimization problems. Similar to the previous Chapter, numerical experiments are conducted to evaluate the performance of the proposed approach for geometric programs. These experiments aim to assess the accuracy and efficiency of the method in solving dependent joint chance-constrained geometric optimization problems in practical scenarios.

Chapter 6 investigates distributionally robust geometric optimization problems. It introduces the topic and focuses on linear programs, proposing a neurodynamic duplex approach to solve these problems. The chapter provides a convergence analysis and conducts numerical experiments to evaluate the performance of the proposed method. Furthermore, the chapter explores distributionally robust geometric programs with uncertainty sets defined by the first two order moments and presents a dynamical recurrent

neural network approach to solve them. Numerical experiments are conducted to validate the effectiveness of this approach. The chapter starts with an introduction to the topic, providing an overview of the challenges and motivations behind distributionally robust optimization. The chapter then focuses on linear programs and presents a neurodynamic duplex approach to solve distributionally robust geometric optimization problems in this context. The proposed approach utilizes neural network architectures and dynamic systems to update the decision variables and achieve convergence iteratively. The chapter provides a detailed convergence analysis, demonstrating the stability and convergence properties of the neurodynamic duplex approach. To assess the performance of the proposed method, the chapter conducts numerical experiments. These experiments solve various distributionally robust linear programs using the neurodynamic duplex approach. The results of the experiments are analyzed to evaluate the accuracy, efficiency, and effectiveness of the proposed method in practical scenarios. In addition to linear programs, the chapter explores distributionally robust geometric programs with uncertainty sets defined by the first two order moments. The chapter presents a dynamical recurrent neural network approach specifically designed to solve these distributionally robust geometric programs. This approach leverages the recurrent neural network architecture to capture the temporal dynamics of the optimization process. Numerical experiments are conducted to validate the effectiveness of the proposed approach for distributionally robust geometric programs. These experiments solve various optimization problems with uncertainty sets defined by the first two order moments using the dynamical recurrent neural network approach. The results of the experiments provide insights into the performance and efficacy of the proposed method in addressing distributionally robust geometric optimization problems.

Chapter 7 serves as a comprehensive summary of the main findings and contributions presented throughout the thesis. It begins by revisiting the research objectives and highlighting the significance of the proposed neurodynamic approaches for solving chance-constrained geometric optimization problems. The chapter then summarizes the key contributions of the research, emphasizing the novel methodologies and techniques developed to address various types of geometric optimization problems. It discusses the advantages of the proposed neurodynamic approaches, such as their ability to handle nonconvexity, address joint probabilistic constraints, and provide robust solutions. Furthermore, the chapter explores the implications and potential applications of the proposed neurodynamic approaches. It discusses how these approaches can be applied in real-world scenarios across different domains, such as transportation, shape optimization, signal processing, and more. The potential benefits of incorporating these approaches into decision-making processes are highlighted, including improved accuracy, efficiency, and robustness of optimization solutions. In the concluding section of the chapter, possible future directions for research in the field of geometric optimization are discussed. These directions may include further investigation into specific types of geometric programs, exploring the integration of additional constraints or objectives, extending the neurodynamic approaches to handle larger-scale problems, or adapting the methodologies for

specific application domains.

2 - Literrature review

2.1 . Geometric Optimization

In his investigation of techniques to minimize costs in engineering design problems, Zener [146, 147] sparked the initial research in what is now known as geometric programming (GP). Zener's work, along with subsequent papers by Duffin and Peterson [37, 38, 39], as well as the book by Duffin, Peterson, and Zener [107], laid the foundational groundwork for this field. The term "geometric programming" was adopted due to the significant role played by the arithmetic-geometric mean inequality in its early development. Initially, the focus of geometric programming was primarily on minimizing posynomial functions while adhering to inequality constraints. Hence, it could have been named posynomial programming instead of geometric programming. Geometric programming offers a means to express and analyze numerous significant optimization problems in a separable format, even when they are typically considered inseparable. The fundamental aspect of this approach lies in leveraging the linear properties inherent in the problem at hand.

2.1.1 . Geometric Optimization: Definition and Concepts

Let x_1, x_2, ..., x_n denote n real positive variable, then a real-valued function f in the form

$$f(x) = cx_1^{\alpha_1}x_2^{\alpha_2}...x_n^{\alpha_n}$$

where $c > 0$ and α_i, $i = 1, ..., n$ are real constants is called *monomial* function. The constant c within the monomial is commonly known as the coefficient, while the constants $\alpha_1, ..., \alpha_n$ are referred to as the exponents, signifying the respective powers of the variables in the monomial. For instance, the monomial $4.3x_1^5x_2^{-0.0005}$ of variables x_1 and x_2 where the coefficient is 4.3 and the exponents are 5 and -0.0005.

A function $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is called a *posynomial*, if it takes the form of a non-negative sum of monomials. The term "posynomial" is coined by combining the words "positive" and "polynomial," implying a mathematical expression that possesses both the properties of positivity and being a polynomial.

$$f(x) = \sum_{i=1}^{n} c_i \prod_{j=1}^{M} x_j^{\alpha_{ij}}$$

$3.2x_1x_2^{0.4} + 1.5x_1^2x_2^{-2}$ and $\sqrt{3}x_1 + x_1x_2^{-2}x_3^{-2}$ are posynmials.

Geometric programming is used to minimize functions that are formulated as posynomials while satisfying constraints of the same posynomial form. A general form of a geometric program can be then stated as follows

$$\min_{x \in \mathbb{R}^M_{++}} \sum_{i \in I_0} c_i \prod_{j=1}^{M} x_j^{\alpha_{ij}}, \quad \text{subject to} \quad \sum_{i \in I_k} c_i \prod_{j=1}^{M} x_j^{\alpha_{ij}} \leq 1, \;\; k = 1, ..., K, \tag{2.1}$$

where x is a strictly positive M-dimensional vector, the exponents α_{ij} are arbitrary real numbers and the coefficients c_i are positive.

Monomials and posynomials are typically non-convex functions. However, it is possible to transform them into convex functions by applying a simple variable transformation. This transformation allows us to convert the original non-convex functions into equivalent convex forms, which can then be effectively optimized using convex optimization techniques. Program (2.1) has then an equivalent convex program given by taking $t_i = \log(x_i)$.

$$\min_t \sum_{i\in I_0} c_i \exp\left(\sum_{j=1}^{M} \alpha_{ij} t_j\right), \quad \text{subject to} \quad \sum_{i\in I_k} c_i \exp\left(\sum_{j=1}^{M} \alpha_{ij} t_j\right) \leq 1, \;\; k = 1, ..., K, \tag{2.2}$$

The equivalence between geometric programs and convex programs has significant implications. One of the most crucial consequences is that the principles and techniques of convex programming can be directly applied to geometric programs, particularly when expressed in their equivalent logarithmic form. This enables the utilization of general theories and algorithms from convex programming to solve geometric programs effectively. Nevertheless, in certain real-world scenarios, it may be impractical to express the problem in its conventional geometric format, rendering it non-convex. Consequently, solving such problems becomes notably challenging, even with approximate methods. In such instances, it becomes highly advantageous to employ straightforward approaches capable of generating a favorable suboptimal solution, if not the optimal solution.

Geometric programming is named as such because it is founded on the principles of the arithmetic-geometric mean inequality. This inequality plays a fundamental role in formulating and solving geometric programming problems. By leveraging the arithmetic-geometric inequality, geometric programming offers a powerful framework for optimizing and solving problems involving posynomials and monomials, ultimately leading to efficient solutions in various domains. In [107], Duffin, Peterson, and Zener employ an extension of the weighted arithmetic-geometric mean inequality to derive lower bounds on the minimum value of a geometric program (2.1). The inequality states that for every $x > 0$ and $y > 0$ vectors in $\mathbb{R}^n$, we have

$$\sum_{i=1}^{n} x_i^{\lambda} \geq \prod_{i=1}^{n} \left(\frac{x_i}{y_i}\right)^{y_i} \lambda^{\lambda}, \tag{2.3}$$

where $\sum_{i=1}^{n} y_i = \lambda$.

We can apply the the arithmetic-geometric mean inequality to the objective function of (2.1), we obtain

$$\sum_{i\in I_0} c_i \prod_{j=1}^{M} x_j^{\alpha_{ij}} \geq \prod_{i\in I_0} \left(\frac{c_i \prod_{j=1}^{M} x_j^{\alpha_{ij}}}{\delta_i}\right)^{\delta_i}, \tag{2.4}$$

with $\sum_{i\in I_0} \delta_i = 1$. The constraints can be expressed as

$$1 \geq \left(\sum_{i\in I_k} c_i \prod_{j=1}^{M} x_j^{\alpha_{ij}}\right)^{\lambda_k} \geq \prod_{i\in I_k} \left(\frac{c_i \prod_{j=1}^{M} x_j^{\alpha_{ij}}}{\delta_i}\right)^{\delta_i} \lambda_k^{\lambda_k}, k = 1, ..., K \tag{2.5}$$

Those inequalities are crucial keys for the duality in geometric programming, more details can be found in [41]. The primal problem (2.1) is often characterized by its complexity, while the dual version of the problem is comparatively simpler to solve. In practice, it is common to focus on solving the dual problem, as it provides valuable insights. The dual problem of (2.1) consists on maximizing the following dual function

$$\prod_{k=0}^{K} \prod_{i\in I_k} \left(\frac{c_i}{\delta_i}\right)^{\delta_i} \prod_{k=1}^{K} \lambda_k^{\lambda_k} \tag{2.6}$$

subject to the following normality and orthogonality conditions

$$\sum_{i\in I_0} \delta_i = 1 \tag{2.7}$$

$$\sum_{k=0}^{K} \sum_{i\in I_k} a_{ij}\lambda_i = 0, j = 1, ..., M. \tag{2.8}$$

where $\sum_{i\in I_k} \delta_i = \lambda_k$.

2.1.2 . Solution Approaches and Algorithms

The solution of geometric programs using numerical methods is a highly interesting topic, and researchers have developed various specialized algorithms for this purpose. A fundamental question arises regarding whether to solve the primal or the dual program. In the case of regular geometric programs, the dual program has a concave objective function and linear constraints, while the primal program typically contains nonlinear constraints. Consequently, it may seem reasonable to solve regular geometric programs through the dual approach. However, in practice, this approach is not always recommended because if certain dual variables need to be zero in the optimal solution, the objective function's gradient becomes unbounded, leading to significant numerical challenges. Moreover, since programs more general than regular geometric programs either lack a dual representation or do not provide apparent computational advantages through the dual, several algorithms based on the primal approach have been developed. Many of these algorithms employ condensation, a technique that combines a sum of positive terms into a single term, enabling the approximation of a nonconvex program with a convex one. There exist several computational methods to solve geometric programs, those methods can be divided into two families : dual methods that solve the dual program, and subsequently, the duality relations are utilized to derive a solution for the primal geometric program and primal methods.

The first work that proposed an algorithm for posynomial programming was the dual method of Frank [50]. Rijckaert and Martens [115] propose a dual condensation algorithm to come up with a solution to generalized geometric programs. Bricker and Rajgopal [16] describe an algorithm for the geometric programming dual problem using an adaptation of the generalized LP algorithm. Kortanek et al. [78] present an infeasible interior-point algorithm for solving primal and dual geometric programs. By employing the Zadeh's extension principle and transforming the primal problem of fuzzy geometric programming into its dual form, Saraj [119] facilitates the conversion of the dual form into a pair of mathematical programs. To achieve this, he applies the α-cut on the objective function and the r-cut on the constraints in the dual form of geometric programming. This process allows us to obtain acceptable (α, r) optimal values, which represent solutions that strike a balance between precision (α) and satisfaction of constraints (r) in the fuzzy geometric programming problem. Rosenberg [118] introduces a novel technique that aims to generate an initial estimate of a primal solution for a posynomial geometric program using a dual feasible point that satisfies a mild condition. It is important to note that while the dual feasible point may not be optimal, this technique provides a means to derive a preliminary primal solution. By leveraging this approach, it becomes possible to obtain a starting point for further optimization iterations, which can subsequently lead to the discovery of improved primal solutions for the posynomial geometric program.

Even if during the initial phase greater emphasis was placed on the development of dual methods, Numerous techniques have been developed to solve the primal posynomial program directly. Dinkel et al. [36] apply four cutting plane algorithms to solve posynomial geometric programs. Chun-Feng et al [30] present an efficient branch and bound algorithm for globally solving the sum of geometric fractional functions under geometric constraints. Melissaratos and Souvaine [98] employ shortest path theory to deal with a class of geometric optimization problems. Sassen et al. [120] use demonstrate the benefits of using nonlinear rotation-invariant coordinates on a collection of geometric optimization problems. Ojha and Ota [103] propose a method that addresses the convexity issues associated with the weighted sum technique and known as the ϵ-constraint method. This approach involves minimizing a primary objective while expressing the remaining objectives as inequality constraints. By treating the additional objectives as constraints, the -constraint method enables the exploration of the trade-offs between the primary objective and the secondary objectives in a more flexible manner. This approach allows for a more nuanced optimization process that considers both the primary objective and the satisfaction of the specified constraints.

The degree of difficulty of GP is defined as the difference between the total number of monomial terms in the objective and constraints, and one plus the number of variables. When the degree of difficulty is zero, solving a GP is equivalent to solving a system of linear equations. Remarkably, modern numerical methods demonstrate good performance regardless of the degree of difficulty. These methods can effectively handle GP problems with varying levels of complexity, making them suitable for a wide range of applications. In this survey, we take an in-depth look at two algorithms that make use of

modern convex optimization techniques, namely the interior-point method [17] and an infeasible Primal-Dual algorithm [79].

- **Barrier-based interior-point method.** The standard barrier-based interior-point method, commonly used for convex optimization, can be directly applied to GP problems. This approach offers a worst-case polynomial-time complexity and exhibits highly efficient performance that gracefully scales with the problem size in practical scenarios. By leveraging the interior-point method, solving GP problems becomes more tractable and efficient, enabling the optimization of larger-scale problems within reasonable computational timeframes. Let us consider the following GP problem formulated in convex form

$$\min_t f_0(x) = \sum_{i \in I_0} c_i \text{exp}\left(\sum_{j=1}^{M} \alpha_{ij} x_j\right), \quad \text{subject to} \quad f_k(x) = \sum_{i \in I_k} c_i \text{exp}\left(\sum_{j=1}^{M} \alpha_{ij} x_j\right) \leq 1, \quad k = 1, ..., K, . \tag{2.9}$$

  The fundamental concept behind the barrier method is to solve a sequence of unconstrained problems by incorporating the constraints into a modified objective function. This new objective function is a weighted combination of the original objective function and a barrier function ϕ that represents the constraints. As the weight parameter t assigned to the original objective function increases, the unconstrained problem progressively approximates the original problem more closely. This iterative process allows for the optimization of the problem while gradually considering the constraints and finding solutions that satisfy them. We can describe the barrier method algorithm as follows

  Given a strictly feasible point x, and given initial values $t_0 > 0$, $\mu > 1$, and an error tolerance $\epsilon > 0$, the following steps are repeated:

  - Compute $x^*(t)$ by minimizing the function $tf_0(x) + \phi(x)$, starting at x. This is an unconstrained, smooth, convex minimization problem that can be solved using iterative methods such as the gradient descent method or Newton's method.
  - Set x to the newly obtained solution $x^*(t)$.
  - If the barrier parameter $\frac{K}{t}$ is less than or equal to the specified error tolerance ϵ, exit the loop.
  - Update t, i.e., $t = \mu t$.

- **The infeasible Primal-Dual algorithm.** The infeasible primal-dual method is an optimization approach that generates subfeasible solutions, meaning solutions that do not satisfy all the constraints exactly. However, the method ensures that the primal and dual objective function values of these subfeasible solutions converge to the respective primal and dual optimal values.

The method involves solving a sequence of optimization problems, starting from an initial infeasible solution. The objective is to improve both the primal and dual objective function values iteratively while gradually satisfying the constraints.

By iteratively adjusting the solution and updating the Lagrange multipliers, the method aims to converge towards a solution that is feasible and has optimal objective function values for both the primal and dual problems.

While the generated solutions may not be strictly feasible, the infeasible primal-dual method provides a way to obtain increasingly better approximations to the primal and dual optimal values as the iterations progress. The details of the algorithm are given in [79].

2.1.3 . Stochastic geometric programming

Traditionally, geometric programming has been applied to problems with known and precise parameter values. However, in real-life applications, the observed values of these parameters are often imprecise or vague. Such uncertain data can take different forms, including bounded ranges, intervals, fuzzy sets, or random variables.

In order to handle these uncertainties, extensions and modifications to the traditional geometric programming approach have been developed. These variations incorporate techniques from interval analysis, fuzzy optimization, stochastic programming, or robust optimization. By considering the uncertain nature of the parameters, these approaches provide more robust and reliable solutions to real-life geometric programming problems.

Therefore, with the incorporation of uncertain or imprecise data, geometric programming becomes a versatile and adaptable tool for solving optimization problems in practical applications. It enables decision-makers to make informed decisions and find optimal solutions even in the presence of uncertainty.

In stochastic geometric programming, it is acknowledged that certain coefficientsc_i and/or exponents a_{ij} in the geometric programming problem are not known precisely, and their incomplete knowledge is represented using probabilistic or random models.

The uncertainty associated with these coefficients can arise from various sources such as measurement errors, variability in data, or inherent randomness in the system being modeled. Instead of assuming deterministic values for these uncertain parameters, stochastic geometric programming incorporates their randomness into the problem formulation.

The origins of stochastic geometric programming can be traced back to the work of Avriela and Wilde [8], where the exponents a_{ij}in the geometric programming problem are deterministic, while the coefficients c_i are considered as positive random variables. In the case of individual probabilistic constraints, the constraints of (2.1) can be replaced by

$$\mathbb{P}\left(\sum_{i\in I_k} c_i \prod_{j=1}^{M} x_j^{\alpha_{ij}} \leq 1\right) \geq 1-\epsilon_k, \quad k=1,....,K \tag{2.10}$$

where $1-\epsilon_k$ are the tolerance levels.

Jagannathan [65] considers the coefficients and the exponents as random variables with known joint distribution functions. This allowed to model the uncertainty associated with these parameters and analyze the properties of the problem under this stochastic setting. The approach consists of a multiplicative recourse stochastic model. In this model, the recourse variables are introduced to rectify or adjust the solution in a proportional manner to account for the possible violations of the constraints due to the randomness in the parameters. Dupačová [40] presents stochastic geometric programming in the context of metal cutting optimization. Liu [92] introduces a procedure that allows for the derivation of lower and upper bounds for the objective function in a posynomial geometric programming problem when there is uncertainty in the cost and constraint parameters. To address this uncertainty, a transformation of the imprecise geometric program into a collection of conventional geometric programs was proposed. Each program within this collection corresponds to a specific combination of parameter values from the provided ranges. By solving these conventional geometric programs, the objective value can be computed. Liu [93] presents a procedure to determine the fuzzy objective value in a fuzzy posynomial geometric programming problem. The fuzziness arises from the exponents of decision variables in the objective function, the cost and constraint coefficients, and the right-hand sides, all of which are represented as fuzzy numbers. The approach is based on Zadeh's extension principle, which allows for the transformation of the fuzzy geometric programming problem into a pair of two-level mathematical programs. By utilizing a duality algorithm and a simple algorithm, this pair of two-level mathematical programs is further transformed into a pair of conventional geometric programs. Liu et al. [88] focus is on geometric programs with joint probabilistic constraints. Specifically, when the stochastic parameters follow independent normal distributions, an approximation approach is proposed. This approach involves approximating the original problem using piecewise linear functions and transforming it into a convex geometric program. Additionally, a sequential convex optimization algorithm is developed to find an upper bound. Khanjani et al. [74] formulate three variants of chance-constrained GP based on different perspectives: possibility, necessity, and credibility. Each variant represents a different way of handling uncertainty and capturing the level of confidence in achieving the desired outcomes.

2.1.4 . Applications of geometric programming

Since its appearance, geometric optimization has found many successful applications in various fields such as mechanical engineering [66, 46, 67, 85, 134], chemical engineering [33, 139], heat exchanger system optimization [10], power control [83, 129], communication network systems [29, 58] and Finance [71].

To showcase the versatility of geometric programming and highlight the formulation of specific models, we will delve into three detailed applications. These applications serve as examples that demonstrate the wide range of domains where geometric programming finds utility.

Design of a Two-bar Truss

To achieve minimum weight, we need to design the two-bar truss, as depicted in Figure 2.1, for a vertical load of $2P$. The truss members have a tubular section with a mean diameter d and wall thickness t. The maximum allowable stress in each member is denoted as σ_s. Using geometric programming, we aim to determine the values of h (height) and d (diameter) considering the following data

- P is the vertical load (33000 lb)
- t is the wall thickness (0.1 in)
- b is the span (30 in)
- σ_s is the maximum permissible stress (60000 psi)
- ρ is the density (0.3 lb/in³)

The objective function for this problem is defined as

$$f(d,h) = 0.188d * 900 + h^2 \tag{2.11}$$

subject to the following stress constraint

$$\frac{33000}{0.1\pi d}\frac{\sqrt{900+h^2}}{h} \leq 60000. \tag{2.12}$$

The stress constraint ensures that the maximum stress is within the permissible limit of 60000. The goal of the geometric programming problem is to find the optimal values of d and h that minimize the objective function while satisfying the stress constraint.

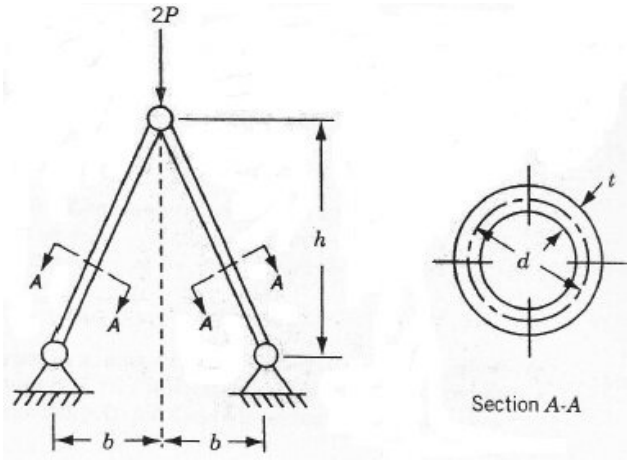

Figure 2.1: Two-bar truss under load [100]

The square root functions in equations (2.11) and (2.12) are not posynomials due to the term $\sqrt{900+h^2}$. However, we can transform these functions into posynomials by introducing a new variable y, defined as $y = \sqrt{900+h^2}$.

The optimization problem becomes

$$\begin{aligned} &\min\ 0.188yd \\ &\text{s.t. } 1.75yh^1d^1 \leq 1, \\ &\quad 900y^2 + y^2h^2 \leq 1. \end{aligned}$$

The corresponding dual problem can be formulated as follows

$$\begin{aligned} &\max \left(\frac{0.188}{\delta_{01}}\right)^{\delta_{01}} \left(\frac{1.75}{\delta_{11}}\right)^{\delta_{11}} \left(\frac{900}{\delta_{21}}\right)^{\delta_{21}} \left(\frac{1}{\delta_{22}}\right)^{\delta_{22}} (\delta_{21}+\delta_{22})^{\delta_{21}+\delta_{22}} \\ &\text{s.t. } \delta_{01} = 1, \\ &\quad \delta_{01} + \delta_{11} - 2\delta_{21} - 2\delta_{22} = 0, \\ &\quad -\delta_{11} + 2\delta_{22} = 0, \\ &\quad \delta_{01} - \delta_{11} = 0. \end{aligned} \tag{2.13}$$

The solutions of (2.13) are given by $\delta_{01}^* = 1$, $\delta_{11}^* = 1$,$\delta_{21}^* = 0.5$ and $\delta_{22}^* = 0.5$. The maximum value of f is then given by $f^* = \left(\frac{0.188}{1}\right)^1 \left(\frac{1.75}{1}\right)^1 \left(\frac{900}{0.5}\right)^{0.5} \left(\frac{1}{0.5}\right)^{0.5} (0.5+0.5)^{0.5+0.5} = 19.8$. We finally find $y^* = 42.426$, $h^* = 30$ in and $d^* = 2.475$ in.

Helical springs design

To determine the minimum volume design of the cone clutch depicted in Figure 2.2 while ensuring the transmission of a specified minimum torque, we can employ the outer and inner radii of the cone, denoted as R_1 and R_2, as design variables. The objective function

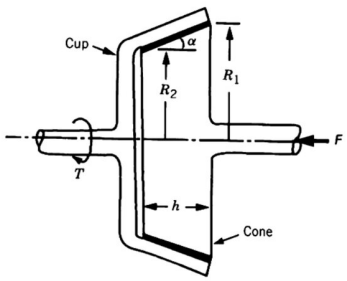

Figure 2.2: Schematic of a cone clutch [80]

can then be formulated as follows

$$f(R_1, R_2) = R_1^3 - R_2^3 \tag{2.14}$$

subject to the following constraints

$$\frac{R_1^2 + R_1R_2 + R_2^2}{R_1 + R_2} \geq 5, \tag{2.15}$$

$$\frac{R_1}{R_2} \geq 2. \tag{2.16}$$

The problem was addressed using an iterative approach known as complementary geometric programming [9]. Consequently, the final solution is determined as$R_1^* = 4.2874$, $R_2^* = 2.1437$ and $f^* = 68.916$.

Production Model Under Risk

In our initial application, we focus on a straightforward profit-maximization model for a single-product firm with predetermined input prices and output price. We then extend the analysis to incorporate stochastic elements, introducing uncertainty into the input prices and output prices, thereby considering the stochastic case. This extension enables us to explore the implications of uncertainty on the firm's profit-maximization strategy within a geometric programming framework. The analysis begins with a simplified scenario where the production of a single output involves the combination of three inputs: labor x_3, raw materials x_2, and capital x_3. The geometric programming formulation for the profit-maximization model in the deterministic case can be represented as follows

$$\min\ 0.14x_1 + 0.04x_2 + 0.06x_3 \tag{2.17}$$
$$\text{s.t. } 10x_1^{-1.6}x_2^{-0.5}x_3^{-0.7} \leq 1, \tag{2.18}$$
$$0.1x_1x_3^{-1} \leq 1, \tag{2.19}$$
$$5x_5^{-1}x_3 \leq 1. \tag{2.20}$$

The dual program can be expressed as follows through

$$\begin{aligned}
\max\ & \left(\frac{0.14}{\delta_1}\right)^{\delta_1}\left(\frac{0.04}{\delta_2}\right)^{\delta_2}\left(\frac{0.06}{\delta_3}\right)^{\delta_3}\left(\frac{10}{\delta_4}\right)^{\delta_4}\left(\frac{0.1}{\delta_5}\right)^{\delta_5}\left(\frac{5}{\delta_6}\right)^{\delta_6} \\
\text{s.t. } & \delta_1 + \delta_2 + \delta_3 = 1, \\
& \delta_1 - 1.6\delta_4 + \delta_5 - \delta_6 = 0, \\
& \delta_2 - 0.5\delta_4 = 0, \\
& \delta_3 - 0.7\delta_4 - \delta_5 + \delta_6 = 0, \\
& \delta_1, \delta_2, \delta_3 \geq 0.
\end{aligned}$$

The optimal solution is 0.6516 and we get
$x_1 = 3.5213$, $x2 = 2.9089$, $x3 = 7043$,
the dual optimal values are
$\delta_1 = 0.75658$, $\delta_2 = 0.17857$, $\delta_3 = 0.06485$ $\delta_4 = 0.35714$, $\delta_5 = 8.427 \times 10^{-8}$ and $\delta_6 = 0.18515$.

As a straightforward approach to incorporate stochastic elements, we can consider a scenario where production needs to align with sales. In this context, sales are treated as a random variable with a normal distribution, with a mean of 1000 units and a variance of

100. With this assumption, our model becomes

$$\min\ 0.14x_1 + 0.04x_2 + 0.06x_3 \tag{2.21}$$
$$\text{s.t. } \mathbb{P}(y \geq y^*) \geq \alpha, \tag{2.22}$$
$$y = 100x_1^{1.6}x_2^{0.5}x_3^{0.7}, \tag{2.23}$$
$$0.1x_1x_3^{-1} \leq 1, \tag{2.24}$$
$$5x_5^{-1}x_3 \leq 1. \tag{2.25}$$

where y^* denotes sales, a random variable with a normal distribution with a mean of $\bar{y} = 1000$ and variance of $\sigma^2 = 100$. The constraint (2.22) indicates that the firm aims to implement a production policy that ensures a high probability of meeting sales targets. The tolerance measure α represents the desired probability threshold.

2.1.5 . Extensions of Geometric Programming

A notable limitation in the effective application of geometric programming to optimize engineering design has been the requirement that all functions involved in the problem must be posynomials. As a result, extensive research efforts have been undertaken to develop extensions and enhancements for geometric programming. These extensions aim to relax the posynomial constraint and allow for a broader range of functions to be incorporated into the optimization framework. These advancements have expanded the applicability and versatility of geometric programming in engineering design optimization.

Sigmonial geometric programming

SGP, signomial geometric programming, represents a significant expansion of GP. Several approaches have been proposed and numerous specialized algorithms have been developed to tackle SGP. However, despite these valuable contributions, the challenge of solving SGP problems remains an open issue. This difficulty primarily arises from the inherent non-convexity of SGP. Unlike GP problems, SGP problems retain their non-convex nature in both the primal and dual forms, and there is no known transformation capable of convexifying them. As a result, efficiently computing only a locally optimal solution for an SGP problem is currently achievable.

SGP is similar to Posynomial GP, with the key distinction that the term coefficients c_i are no longer required to be positive. In Posynomial GP, all coefficients must be positive, whereas in SGP, the coefficients can have both positive and negative values. A general form of a signomial program can be stated as follows

$$\begin{aligned} \min_{x\in\mathbb{R}^M_{++}}\ & \sum_{i=1}^{I_0}\sigma_i^0 c_i^0 \prod_{j=1}^{M} x_j^{a_{ij}^0}, \\ \text{s.t}\ & \sum_{i=1}^{I_k}\sigma_i^k c_i^k \prod_{j=1}^{M} x_j^{a_{ij}^k} \leq \sigma_{k0}, k = 1, ..., K, \\ & c_i^k > 0, \sigma_i^k, \sigma_{k0} = +1 \text{ or } -1. \end{aligned} \tag{2.26}$$

The dual program is given by

$$\begin{aligned}
\max\ & \sigma_{00} \left(\prod_{k=0}^{K} \prod_{i \in I_k} \left(\frac{c_i}{\delta_i} \right)^{\sigma_i^k \delta_i} \prod_{k=1}^{K} \lambda_k^{\lambda_k \sigma_{k0}} \right), \\
\text{s.t}\ & \sigma_{k0} \sum_{i \in I_k} \sigma_i^k \delta_i = \lambda_k, k = 1, ..., K, \\
& \sum_{k=0}^{K} \sum_{i \in I_k} a_{ij} \delta_i \sigma_i^k = 0, j = 1, ..., M, \\
& \sigma_{00} \sum_{i \in I_0} \sigma_i^0 \delta_i = 1, \\
& \delta_i, \lambda_k \geq 0,
\end{aligned} \tag{2.27}$$

where σ_{00} represents the sign of the primal objective at the optimum.

There exist a lot of approaches to solve SGP, we can state the following three major techniques

- **Branch and bound:** The first approach involves employing standard branch and bound techniques for general non-convex optimization. Gochet and Smeers [53] propose a branch-and-bound method for solving SGP, ensuring convergence to the global optimum. The subproblems involved in this method are convex and were combined with a cutting plane technique to generate linear subproblems. Shen and Jiao [122] propose an algorithm that successfully integrates the branch-and-bound method with the pruning technique, leading to convergence to the global minimum. This achievement is made possible by iteratively refining the linear relaxation of the feasible region of the objective function and solving a series of linear relaxation problems. Through this process of successive refinement and solving relaxed problems, the algorithm progressively approaches the global minimum.

- **Relaxations:** Shen and Zhang [123] use an exponential variable transformation, tangential hypersurfaces and convex envelope approximations to obtain a linear relaxation of SGP. Consequently, the initial nonconvex nonlinear problem of SGP is transformed into a sequence of linear programming problems via the successive improvement of a linear relaxation of the objective function's feasible region. By iteratively solving this series of linear programming problems, the proposed algorithm demonstrates convergence to the global minimum of SGP. Chandrasekaran and Shah [18] use a hierarchy of convex relaxations to obtain progressively tighter lower bounds on the optimal value of SGPs. These lower bounds are obtained by solving increasingly larger relative entropy optimization problems, which are convex programs formulated using linear and relative entropy functions. The key insight behind this approach is the observation that the relative entropy function, due to its joint convexity with respect to both arguments, offers a convex parameterization for specific sets of globally nonnegative signomials. This parameterization

allows for efficiently computable nonnegativity certificates using the arithmetic-geometric-mean inequality.

- **Reversed GP:** The third approach involves converting an SGP problem into a Reversed GP problem and then applying iterative monomial approximation. Shen et al. [124] minimize a SGP problem by converting it into a reversed GP problem, which is then solved using standard GP solvers. Aliabadi et al. [6] consider a nonlinear model that incorporates joint partial delayed payments, pricing, and marketing strategies in a supply chain consisting of a retailer and multiple customers. The problems are formulated using a constrained SGP with two levels of difficulty. To solve the proposed models, the researchers transform them into a reversed constraint programming framework and obtain optimal solutions in closed forms for each case.

As an example, we can take the problem of maximizing the total capacity of the wireless network from [59], which can be stated as follows

$$\max_{p\in\mathcal{K}_{++}^{K}} \min_{i\in\mathcal{U}} log_2\left(1+\frac{p_i|g_i^H g_i|^2}{\sum\limits_{j\in\mathcal{U},j\neq i} p_j|g_i^H g_j|^2+|\sigma_i|^2}\right), \tag{2.28}$$

$$\text{s.t} \quad P_{min}\leq p_i\leq P_{max}, \forall i\in\mathcal{U}, \tag{2.29}$$

By taking $a_{ij}=|g_i^H g_j|^2$ and $b_i=|\sigma_i|^2$, we obtain the following signomial programming problem [4]

$$\max_{p,t\in\mathcal{K}_{++}^{2K}} \prod_{i\in\mathcal{U}} t_i, \tag{2.30}$$

$$\text{s.t} \quad t_i\leq 1+\frac{a_{ii}p_i}{\sum\limits_{j\in\mathcal{U},j\neq i} a_{ij}p_j+b_i}, \forall i\in\mathcal{U}, \tag{2.31}$$

$$P_{min}\leq p_i\leq P_{max}, \forall i\in\mathcal{U}. \tag{2.32}$$

The search for the global optimal solution of problem (2.30)-(2.32) is challenging due to its non-convexity. However, in an effort to find the global optimal solution, one can still employ any of the aforementioned methods mentioned earlier to solve the problem.

Quasi geometric programming

Quasi geometric programming (QGP) is a special class of non-convex optimization that can be transformed into geometric programming by holding certain variables constant. The exploration of this specific type of optimization problem, which can be nonlinear and possibly non-smooth, is motivated by the observation that numerous engineering problems can be effectively formulated or approximated as QGPs. Nevertheless, solving a QGP remains challenging due to its inherent non-convex nature. This extended version of GP was introduced in 2010 by Toscano and Lyonnet [133].

A nonlinear optimization problem is referred to as a quasi-geometric programming problem if it can be formulated in the following form

$$\begin{aligned} \min_{x\in^{p}_{++},\zeta\in^{q}_{++}} \quad & \phi_0(x,\zeta) - Q_0(\zeta), \\ \text{s.t} \quad & \phi_k(x,\zeta) \leq Q_k(\zeta), k = 1, ..., K, \end{aligned} \tag{2.33}$$

where the function $\phi_k(x,\zeta)$, $i = 0, 1, ..., K$ are posynomials. Problem (2.33) is called quasi geometric programming in posynomial form if $Q_k(\zeta)$, $k = 0, 1, ..., K$ are ratios of posynomial functions. Otherwise, if $Q_k(\zeta)$, $k = 0, 1, ..., K$ are just assumed to be positive problem (2.33) is called quasi geometric programming in general form. It is crucial to note that in both of these cases, problem (2.33) cannot be converted into a standard form GP, and as a result, the problem is non-convex. Consequently, there is no approach available to swiftly find even a suboptimal solution using existing convex solvers. However, specific algorithms can be designed to discover suboptimal solutions for problem (2.33) in both posynomial or general form.

We consider as example the following optimization problem from [113]

$$\begin{aligned} \min \; & 0.5x_1x_2^1 - x_1 - 5x_2^1, \\ \text{s.t} \; & 0.01x_2x_3^1 + 0.01x_2 + 0.0005x_1x_3 \leq 1, \\ & 70 \leq x_1 \leq 150, 1 \leq x_2 \leq 30, 0.5 \leq x_3 \leq 21. \end{aligned} \tag{2.34}$$

Problem 2.34 can be rewritten as a quasi geometric program (2.35) and can be solved using a global optimization algorithm via Lagrangian relaxation.

$$\begin{aligned} \min \; & \lambda^{-1} \\ \text{s.t} \; & \frac{\lambda + 0.5x_1x_2^1}{x_1 + 5x_2^1} \leq 1, \\ & 0.01x_2x_3^1 + 0.01x_2 + 0.0005x_1x_3 \leq 1, \\ & 70 \leq x_1 \leq 150, 1 \leq x_2 \leq 30, 0.5 \leq x_3 \leq 21. \end{aligned} \tag{2.35}$$

2.2 . Chance Constrained Optimization

In today's complex and dynamic world, decision-makers face a multitude of uncertainties and risks that can significantly impact the outcomes of their choices. Traditional optimization techniques often assume deterministic inputs and fail to capture the inherent variability and randomness present in many real-world scenarios. Chance constrained optimization, also known as probabilistic optimization or stochastic programming, offers a principled approach to handle uncertainties and make decisions that are robust and reliable. In this section, we provide an extensive survey on chance-constrained programming, including its concepts, results, and applications. We start by delving into the fundamental concepts underlying chance-constrained programming, offering a comprehensive discussion on the topic. Following that, we summarize the established results for various

chance-constrained models. Subsequently, we shift our focus to conducting a survey of real-world applications where chance-constrained programming has been employed.

Chance constrained optimization was introduced for the first time by Charnes, Cooper and Symonds [22]. Since then, it has been applied to several real-world problems as aircraft systems [32], machine learning [7, 2], water distribution networks [81, 82], radiation therapy [95], power systems [116], critical infrastructure protection [125] and supply chain finance [135].

A general form of a chance-constrained optimization problem is given below.

$$\min_{x \in X} f(x), \quad \text{subject to} \quad \mathbb{P}\left(h(x,\zeta) \leq 0\right) \geq p, \tag{2.36}$$

where f is the objective function, $h = (h_1, h_2, ..., h_m)$ is an m-dimensional function, $X \in \mathbb{R}^n$ is a deterministic set, x is the decision variable, ζ is a random variable and $p \in [0,1]$ is a given probability level.

2.2.1 . Individual and joint chance constraints

The constraint in (2.36) is called a joint chance constraint since the constraints $h_1(x,\zeta) \leq 0$, $h_2(x,\zeta) \leq 0$,..., $h_m(x,\zeta) \leq 0$, have to be satisfed simultaneously with probability p. Alternatively, each of the following m constraints can be referred to as an individual chance constraint

$$\mathbb{P}\left(h_1(x,\zeta) \leq 0\right) \geq p_1, \mathbb{P}\left(h_2(x,\zeta) \leq 0\right) \geq p_2, ..., \mathbb{P}\left(h_m(x,\zeta) \leq 0\right) \geq p_m \tag{2.37}$$

. Chance constraints were initially formulated as individual chance constraints [22], and further advancements were made in [19, 20, 21]. The concept of Joint Chance Constraints was first introduced by Miller and Wagner [99] where they explore the mathematical properties of chance constrained programming problems, specifically focusing on situations where the constraint is on the joint probability of a multivariate random event. In the early 1970s, Prékopa made significant contributions [109, 110], including the development of comprehensive convexity results for the feasible set of (2.36).

Generally, joint chance constraints are more adequate to use especially when multiple inequalities are used to define a system they lead to more robust solutions. Nevertheless, they are generally more difficult to solve. There exist several methods enabling to convert joint constraints to individual ones.

A joint chance constrained can be written as a set of individual constraints. Using Boole's inequality we have

$$\mathbb{P}\left(\bigcup_{i=1}^{m} h_i(x,\zeta) \leq 0\right) \leq \sum_{i=1}^{m} \mathbb{P}\left(h_i(x,\zeta) \leq 0\right). \tag{2.38}$$

If $\sum_{i=1}^{m} p_i \leq p$, then a feasible solution to (2.37) is also a feasible solution to the joint constraint in (2.36). However, the choice of parameters p_i is generally a difficult exercise. Insightful observations regarding the limitations of this approach can be found in [23].

Tractable approaches for addressing joint chance constrained programming problems can be categorized into two groups : sampling-based approximations and analytical safe approximations. Luedtke and Ahmed [97] study sample approximations of optimization problems with probabilistic constraints. Jeff Hong et al. [61] propose sequential convex approximations to joint chance constraints based on a Monte Carlo approach. Blackmore et al. [15] present a novel method for chance-constrained predictive stochastic control of dynamic systems. The method approximates the distribution of the system state using a finite number of particles. Li et al. [84] present a safe approximation to joint chance-constrained programming, where the constraint functions are additively dependent on a normally-distributed random vector.

2.2.2 . An overview of solutions to chance-constrained optimization

Chance-constrained optimization models often present computational challenges, making them difficult to solve efficiently. Various existing approaches have been developed to address this issue, including decomposition methods, sample average approximations, and the formulation of deterministic equivalents.

Decomposition methods

Luedtke [96] introduces a novel approach for the exact solution of chance-constrained mathematical programs that involve discrete distributions with finite support and random polyhedral constraints. The proposed approach combines decomposition techniques and integer programming methods to handle the complexity of the problem. Bai et al. Adam et al. [3] addresse the problem of stochastic programs with joint chance constraints that involve discrete random distributions and propose a reformulation technique that introduces auxiliary variables. By adding these variables, they transform the original problem into a new formulation. The transformed problem, however, has a non-regular feasible set, which can pose challenges in finding optimal solutions. To address this issue, the authors propose a regularization approach. This involves expanding the feasible set by introducing additional constraints, which increases the solution space. To solve the regularized problem, the authors employ an iterative approach. They solve a master problem iteratively while incorporating Benders' cuts from a slave problem. This iterative process helps refine the solution and improve the convergence towards an optimal solution. [11] present an augmented Lagrangian decomposition method that aims to find high-quality feasible solutions for complex optimization problems, including nonconvex chance-constrained problems. The proposed method differs from existing augmented Lagrangian approaches by allowing randomness to appear in both the left-hand-side matrix and the right-hand-side vector of the chance constraint. The method involves solving a convex subproblem and a 0-1 knapsack subproblem at each iteration. The special structure of the chance constraint enables the efficient computation of the 0-1 knapsack problem in quasi-linear time, which helps keep the computation for discrete optimization subproblems at a relatively low level.

Sample average approximations

The ample average approximation (SAA) method involves generating a set of scenario samples from the underlying random distribution and computing the empirical average of the objective function and constraints based on these samples. Chunlin and Liu [31] propose the use of Conditional Value at Risk (CVaR) to approximate the chance constraint in stochastic programming problems. Specifically, the chance constraint is approximated by its CVaR counterpart, which provides a measure of the expected loss beyond a certain confidence level. To solve the resulting chance-constrained stochastic programming problem, the authors employ the SAA method. Cheng et al. [28] introduce a novel sampling-based method to solve chance constrained programs, which offers several advantages over traditional approaches. The key innovation is that the proposed method formulates an approximation problem that involves only continuous variables, as opposed to the standard SAA formulation that includes binary variables. Kleywegt et al. [76] investigate a Monte Carlo simulation-based approach for solving stochastic discrete optimization problems. The fundamental concept behind such methods is to generate a random sample and approximate the expected value function using the corresponding sample average function. By solving the resulting sample average optimization problem, the approach aims to approximate the optimal solution of the original stochastic problem. This process is iterated multiple times until a predefined stopping criterion is met.

Deterministic equivalent formulations

Nemirovski and Shapiro [102] address the problem of chance constrained optimization, where the objective is to minimize a convex function subject to a set of convex constraints that are randomly perturbed. The goal is to develop a computationally tractable approximation of this problem, specifically a deterministic optimization program that can be efficiently solved and whose feasible set is contained within the original chance constrained problem. A class of convex conservative approximations was proposed for the chance constrained problem. These approximations provide a computationally tractable representation of the problem while ensuring that the feasible solutions remain within the original chance constrained set. Xie and Ahmed [143] investigate deterministic reformulations for joint chance constraints and explores the conditions under which these reformulations are convex. Gopalakrishnan et al. [55] present a novel algorithm that efficiently solves a class of chance-constrained optimization problems under nonparametric uncertainty. The algorithm leverages the concept of Reproducing Kernel Hilbert Space (RKHS) to represent arbitrary distributions as functions allowing to formulate the chance-constrained optimization problem as a minimization of the distance between a desired distribution and the distribution of the constraint functions within the RKHS framework.

2.2.3 . Application : Chance Constrained Portfolio Problem Description

Portfolio optimization is a widely researched problem in the field of finance. It involves selecting the optimal allocation of investments among a set of K possible companies. The

objective is to maximize the revenue generated from these investments.

In the literature, various formulations of the portfolio optimization problem have been studied. One common approach is the risk-averse formulation, where the goal is to minimize the variance of the total return rate. Some recent applications and methods can be found in [45, 42, 24, 130].

In our model, we make certain assumptions about the problem. These assumptions will serve as the foundation for our analysis and optimization approach. We suppose that we have exact knowledge of the return rate distributions. Additionally, there are no extra costs associated with making investments, allowing for flexibility in allocating funds among companies without incurring differential costs. We also include a risk-free investment option with a return rate of 0. To address risk aversion, we formulate the problem using chance constraints, ensuring that the potential loss does not fall below a certain threshold. Our objective is to maximize the expected revenue. We consider a system represented by a sequence of random variables R_i that represent the return rates of K possible investments, and our aim is to find the optimal solution to this system.

$$\max \sum_{i=1}^{K} \mathbb{E}[R_i] x_i, \tag{2.39}$$

$$\text{s.t} \quad \mathbb{P}\left(\sum_{i=1}^{K} R_i x_i \leq -\eta\right) \leq \epsilon \tag{2.40}$$

$$\sum_{i=1}^{K} x_i \leq 1, x_i \geq 0, i = 1, ..., K. \tag{2.41}$$

Under the assumption that $\eta > 0$ and $\epsilon < 0.5$, finding a solution to the aforementioned problem allows us to establish an investment strategy that maximizes the expected return rate while ensuring the probability of experiencing losses exceeding η remains below ϵ. By optimizing the portfolio allocation based on these criteria, we aim to strike a balance between maximizing potential returns and managing risk within the specified threshold.

We consider that the return rates of investments follow a normal distribution and we have complete knowledge of the mean μ_i and standard deviation σ_i for all distributions, i.e., $R_i \sim \mathcal{N}(\mu_i, \sigma_i)$ for $i = 1, ..., K$. Additionally, we assume that the return rates of different investments are independent of each other. We have then

$$\sum_{i=1}^{K} R_i x_i \sim \mathcal{N}\left(\sum_{i=1}^{K} \mu_i x_i, \sqrt{\sum_{i=1}^{K} \sigma_i^2 x_i^2}\right). \tag{2.42}$$

We have then $\mathbb{P}\left(\sum_{i=1}^{K} R_i x_i \leq -\eta\right) = \mathbb{P}\left(\frac{\sum_{i=1}^{K} \mu_i x_i - \sum_{i=1}^{K} R_i x_i}{\sqrt{\sum_{i=1}^{K} \sigma_i^2 x_i^2}} \geq \frac{\eta + \sum_{i=1}^{K} R_i x_i}{\sqrt{\sum_{i=1}^{K} \sigma_i^2 x_i^2}}\right) = 1 - \phi\left(\frac{\eta + \sum_{i=1}^{K} \mu_i x_i}{\sqrt{\sum_{i=1}^{K} \sigma_i^2 x_i^2}}\right)$,
where ϕ is the cumulative probability function of the standard normal distribution. Con-

straint (2.40) can be written equivalently as

$$1-\phi\left(\frac{\eta+\sum\limits_{i=1}^{K}\mu_i x_i}{\sqrt{\sum\limits_{i=1}^{K}\sigma_i^2 x_i^2}}\right)\leq\epsilon, \tag{2.43}$$

which is finally equivalent to

$$\phi^{-1}(1-\epsilon)\sqrt{\sum_{i=1}^{K}\sigma_i^2 x_i^2}-\sum_{i=1}^{K}\mu_i x_i\leq\eta. \tag{2.44}$$

Problem (2.39)-(2.41) becomes then

$$\max\sum_{i=1}^{K}\mu_i x_i, \tag{2.45}$$

$$\text{s.t}\quad \phi^{-1}(1-\epsilon)\sqrt{\sum_{i=1}^{K}\sigma_i^2 x_i^2}-\sum_{i=1}^{K}\mu_i x_i\leq\eta, \tag{2.46}$$

$$\sum_{i=1}^{K}x_i\leq 1, x_i\geq 0, i=1,...,K. \tag{2.47}$$

Problem (2.45)-(2.47) being a convex second-order cone program implies that it can be solved using existing methods developed for solving convex optimization problems.

In the upcoming Chapter, we will provide a comprehensive theoretical overview of neurodynamic neural networks and biconvex optimization. This detailed exploration aims to enhance the understanding of the thesis by establishing a solid foundation in these key areas of study.

3 - Neurodynamic Neural Networks and Biconvex Optimization

3.1 . Dynamical Neural Networks

3.1.1 . Introduction

Throughout one's lifetime, the brain's networks are constantly abuzz with waves of activity. Countless signals harmonize and oscillate, forming the underlying functional architecture that shapes every facet of existence. Emotions, stress levels, musical preferences, aspirations, and dreams all stem from the dynamic behavior of these networks, complemented by adaptive memory systems that strive to better align with one's ever-changing surroundings.

The vast scale and intricate integration of networks in the human brain present formidable challenges in understanding whether traditional forms of computation occur within them and how. Evidence points to information processing taking place at various levels, encompassing modulatory proteins within individual cells, cortical microcircuits, and expansive functional networks spanning the entire brain. However, the experimental grasp of the brain's workings advances slowly. Thankfully, resourceful engineers have devised algorithms that partially simulate aspects of these networks. While no single model can fully encapsulate the sheer complexity and behavior of the human brain, these tools offer researchers a valuable window into how information might be computed and ultimately represented within the activity of distributed networks.

A significant category of logical problems that stem from real-world scenarios can be framed as optimization problems, which can be qualitatively understood as a quest for the optimal solution. These problems are prevalent in fields such as engineering and commerce, as well as in perceptual challenges that require swift resolution by nervous systems.

The fundamental concept behind the neural network approach for optimization is to create a nonnegative energy function and establish a dynamic system that mimics an artificial neural network. This dynamic system typically takes the form of first-order ordinary differential equations. The objective is for the dynamic system to converge towards a static state or equilibrium point, which corresponds to the solution of the underlying optimization problem, starting from an initial point. One notable advantage of using neural networks for solving optimization problems is their hardware implementability. In other words, these neural networks can be implemented using integrated circuits. Moreover, neural networks designed for circuit implementation offer real-time processing capabilities, further enhancing their practical applicability.

In 1985, Hopfield and Tank introduced the first neural network model for solving the Traveling Salesman Problem [62]. Since then, numerous neural network models have been proposed and utilized in both linear programming and nonlinear programming do-

mains. Kennedy and Chua [72] extend the Tank-Hopfield network by incorporating a finite penalty parameter into a neural network framework. This extension demonstrated the versatility and adaptability of neural networks in addressing a broader range of optimization problems beyond the initial applications of the Tank-Hopfield network. In the pursuit of finding exact solutions, researchers have continued to develop additional neural network models for optimization. These new neural network approaches build upon the foundation established by earlier models and aim to improve the accuracy and efficiency of optimization solutions.

Hosseini et al. [63] propose a penalty-based recurrent neural network designed to address a specific class of constrained optimization problems featuring generalized convex objective functions. The proposed model is characterized by a straightforward structure represented through a differential inclusion. It is applicable to a wide range of nonsmooth optimization problems that incorporate affine equality and convex inequality constraints. Notably, the objective function should be regular and pseudoconvex within the problem's feasible region for successful application. Qin and al. [112] present a neurodynamic approach that utilizes a recurrent neural network to solve convex optimization problems with general constraints. The key contribution of the proposed approach is that for any initial point, the state of the neural network reaches the constraint set within a finite time and ultimately converges to an optimal solution of the convex optimization problem. Qin and Xue [111] introduce a novel approach to tackle nonsmooth convex optimization problems with convex inequality and linear equality constraints using a two-layer recurrent neural network. In contrast to existing neural network models, the proposed approach offers several advantages. Firstly, it exhibits a low model complexity, making it computationally efficient and easier to implement. Additionally, the proposed neural network does not rely on penalty parameters, eliminating the need for parameter tuning and improving the overall simplicity of the optimization process. Liu and Zhou [94] present a one-layer recurrent network designed specifically for solving non-smooth convex optimization problems subject to linear inequality constraints. Liu and al. [90] introduce a one-layer recurrent neural network as a solution for pseudoconvex optimization problems that involve linear equality and bound constraints. In comparison to existing neural networks used for optimization, such as projection neural networks, the proposed model exhibits enhanced capabilities in solving a broader range of pseudoconvex optimization problems with equality and bound constraints. Furthermore, it can handle constrained fractional programming problems as a special case.

Recurrent neural networks have proven to be highly effective in solving a wide range of real-world problems. Liu et al. [90] propose a decision-making model based on a recurrent neural network for dynamic portfolio optimization. Yan et al. [144] use recurrent neural networks to address the problem of electro-hydraulic braking (EHB) force allocation for electric vehicles. Kerboua and Kelaiaia [73] employ recurrent neural networks for the health prognosis of wind turbine operations using vibration time series data. The study explores various memory cell variations, including Long Short Time Memory (LSTM), Bilateral LSTM (BiLSTM), and Gated Recurrent Unit (GRU), integrated into different net-

work architectures. Urias et al. [136] design a recurrent neural network for optimization in the context of optimal operation of an electrical microgrid. The microgrid is connected to the utility grid and includes various components such as batteries for energy storage and supply, as well as an electric car. The main objective of the proposed neural network is to determine the optimal power allocation over a one-week time horizon for wind, solar, and battery systems, taking into account the electric car, with the goal of minimizing the power obtained from the utility grid and maximizing the power generated from renewable energy sources.

The dynamical recurrent neural networks are based on ODE systems. We present in the following subsection some necessary theoretical results to study the stability of ODE systems.

3.1.2 . Stability in differential equations

Let consider the following differential equation

$$\dot{x}(t) = f(x(t)),\ x(t_0) = x0 \in \mathbb{R}^n. \tag{3.1}$$

We present some classical results that establish the existence and uniqueness of a solution for (3.1) [114].

Theorem 1. Assuming that f is a continuous mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$, the following conclusions can be made regarding the existence and uniqueness of solutions to equation (3.1).

- Existence: For any given $t_0 \geq 0$ and initial condition $x_0 \in \mathbb{R}^n$, there exists a local solution $x(t)$ defined on an interval $[t_0, s)$ for some $s > t_0$. This means there is a time interval where the solution exists and is well-defined.
- Uniqueness: If f is locally Lipschitz continuous at x_0, then the solution to equation (3.1) is unique within the local interval $[t_0, s)$. This means there is only one possible solution for a given initial condition in this interval.
- Maximal Interval of Existence: The maximal interval of existence, denoted by $[t_0, s(x_0))$, is the largest time interval on which a local solution exists for a given initial condition x_0. If a local solution cannot be extended to a larger interval $[t_0, s)$, then the local interval $[t_0, s)$ is considered the maximal interval of existence.

Theorem 2. Assuming that f is a continuous mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$, if $x(t), t \in [t_0, s(x_0))$, is a maximal solution such $s(x_0) > \infty$, then it implies the following

$$\lim_{t \longrightarrow s(x_0)} ||x(t)|| = +\infty. \tag{3.2}$$

Definition 1. A point $x^* \in \mathbb{R}^n$ is called an equilibrium point of (3.1) if $f(x^*) = 0$.

Definition 2. Let $x(t)$ be a solution of equation (3.1). An isolated equilibrium point x^* is Lyapunov stable if, for any initial condition $x_0 = x(t_0)$ and any scalar $\epsilon > 0$, there exists a $\delta > 0$ such that if $||x(t0) - x*|| \leq \delta$, then $||x(t) - x*|| \leq \epsilon$ for all $t \geq t_0$.

In other words, if we can find a small enough neighborhood around the equilibrium point such that any initial condition within that neighborhood will result in the solution staying within another specified neighborhood around the equilibrium point for all future time, then the equilibrium point is said to be Lyapunov stable. This stability condition ensures that the solution does not diverge too far from the equilibrium point for sufficiently small perturbations in the initial condition.

Definition 3. An isolated equilibrium point x^* is said to be asymptotically stable if in addition to being Lyapunov stable, it also has the property that if $||x(t0) - x * ||\leq \delta$ then $\lim_{t \longrightarrow \infty} x(t) = x^*$.

In other words, if we can find a neighborhood around the equilibrium point such that any initial condition within that neighborhood will result in the solution approaching the equilibrium point as time goes to infinity, then the equilibrium point is said to be asymptotically stable. This stability property ensures that the system converges to a stable state over time.

Definition 4. Let $\Omega \subseteq \mathbb{R}^n$ be an open neighborhood of $\tilde{x}$. A continuously differentiable function $\eta : \mathbb{R}^n \longrightarrow \mathbb{R}$ is said to be a Lyapunov function at the state $\tilde{x}$ (over the set Ω) for the differential equation (3.1) if the following conditions hold

- $\eta(\tilde{x}) = 0$, $\eta(x) > 0$ for $x \in \Omega$, $x \neq \tilde{x}$.
- $\frac{d\eta(x(t))}{dt} \leq 0$, for all $x \in \Omega$

Remark 1. The above conditions define the Lyapunov function and characterize it as an energy function for the system described by the differential equation (3.1). The Lyapunov function provides a measure of the system's energy or potential, and its derivative describes the change in energy over time. By analyzing the behavior of the Lyapunov function and its derivative, we can determine the stability properties of the system and assess whether the equilibrium point x* is stable or not.

Theorem 3. We have the following results

- An isolated equilibrium point x^* is Lyapunov stable if there exists a Lyapunov function over some neighborhood Ω of x^*.
- An isolated equilibrium point x^* is asymptotically stable if there exists a Lyapunov function over some neighborhood Ω of x^* satisfying $\frac{d\eta(x(t))}{dt} < 0$, for all $x(t) \in \Omega$, $x(t) \neq x^*$.

The neurodynamic system approach is a powerful technique for tackling optimization problems. It leverages artificial recurrent neural networks to convert these problems into dynamic systems represented by first-order differential equations. By initializing the system at a specific point, it is expected to converge towards a static state or equilibrium point, which corresponds to the solution of the original optimization problem.

One notable advantage of neural networks for optimization problems is their parallel information processing capability. Neural networks can simultaneously evaluate multiple inputs and compute corresponding outputs, thanks to their inherent parallel structure. This parallelism enables efficient and concurrent information processing, resulting in faster optimization performance than sequential algorithms.

Furthermore, neural networks designed for optimization problems can be implemented in hardware using specialized technologies like integrated circuits or dedicated processing units. This hardware implementation takes advantage of the parallel nature of neural networks, further enhancing their computational speed and efficiency. By utilizing dedicated hardware resources, neural networks can be deployed in real-time applications or embedded systems, enabling efficient and rapid optimization across diverse domains.

To better comprehend the application of recurrent neural networks in optimization, we present two recurrent neural networks from the literature in the coming subsections to solve convex nonlinear optimization problems.

3.1.3 . A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints

Consider the following nonlinear convex program.

$$\begin{aligned} &\min\ f(x), \\ &\text{s.t}\ \ c(x) \leq 0, x \geq 0. \end{aligned} \tag{NCP}$$

where $x \in \mathbb{R}^n$, f is convex and twice differentiable and $c(x) = [c_1(x), c_2(x), ..., c_m(x)]^T$ is an m-dimensional vector-valued continuous function, where each component is convex and twice differentiable. A vector x is considered a feasible solution if it satisfies the constraints of (NCP). In particular, a feasible solution is deemed a regular point if the gradients of the objective function components, and the constraint functions, are linearly independent.

if a feasible regular point x exists, the Karush-Kuhn-Tucker (KKT) conditions for (NCP) can be expressed as follows

$$y \geq 0, c(x) \leq 0, x \geq 0, \tag{3.3}$$
$$\nabla f(x) + \nabla c(x) y = 0, y^T c(x) = 0. \tag{3.4}$$

By utilizing the optimization projection technique, we can establish a relationship between the solution to (NCP) and the solution to a projection formulation given by

$$(x - \alpha(\nabla f(x) + \nabla c(x) y))_+ - x = 0, \tag{3.5}$$
$$(y + \alpha c(x))_+ - y = 0. \tag{3.6}$$

where $(x)_+ = [(x_1)_+, ..., (x_n)_+]$, $(x_i)+ = \max(x_i, 0)$ and α is a positive constant.

Theorem 4. [142] Let x be a feasible and regular point of (NCP). Then, x is an optimal solution to (NCP) if and only if there exists a nonzero m-dimensional vector y such that for any $\alpha \geq 0$ (3.5)-(3.6) holds for (x, y).

Based on the projection formulation presented in equation (3.5)-(3.6), a recurrent neural network model for solving (NCP) is proposed. The following dynamical equation describes the model [142].

$$\frac{dx}{dt} = -x + (x - \alpha(\nabla f(x) + \nabla c(x)y))_+ , \tag{3.7}$$

$$\frac{dy}{dt} = -y + (y + \alpha c(x))_+. \tag{3.8}$$

The dynamical equation represented by equations (3.7)-(3.8) can be implemented using a one-layer recurrent neural network, as depicted in Figure 3.1.

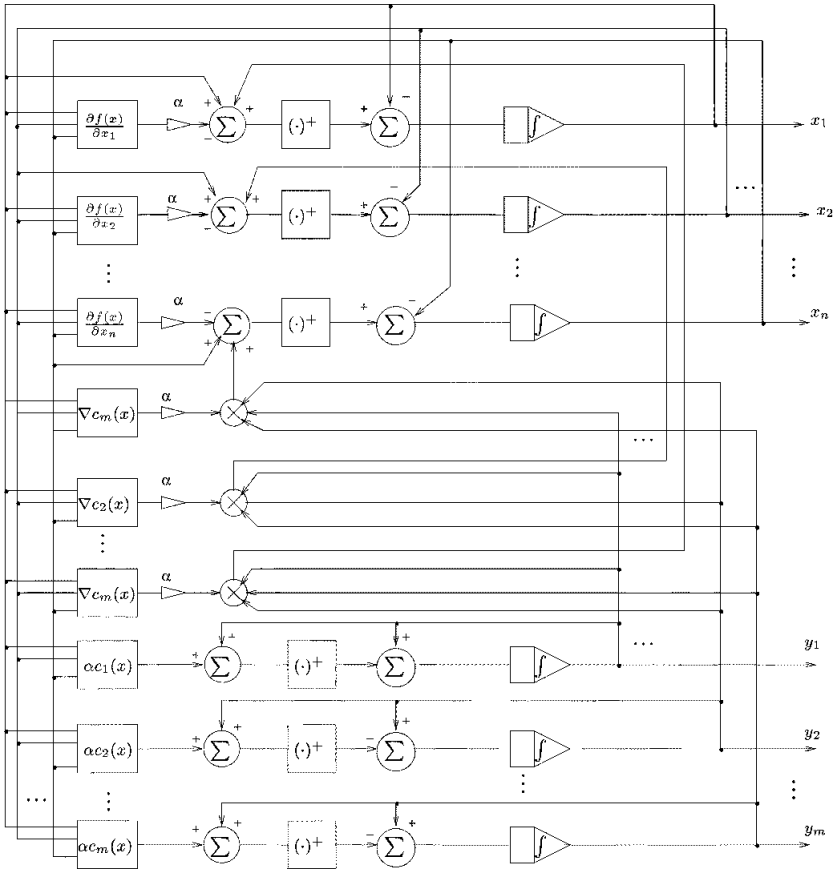

Figure 3.1: Block diagram of neural network (3.7)-(3.8) [142]

Lemma 5. [142] For any initial point $(x(t_0), y(t_0))$, there exists a unique continuous solution $(x(t), y(t))$ for the dynamical system (3.7)-(3.8). Moreover, this solution converges to an equilibrium point that satisfies the projection formulation (3.5)-(3.6).

Theorem 6. [142] If $\nabla^2 f(x) + \sum_{i=1}^{m} y_i \nabla^2 c(x)$, then the neural network described by system (3.7)-(3.8) is globally convergent to a KKT point (x^*, y^*) where x^* corresponds to the optimal solution of the (NCP).

3.1.4 . A dynamic system model for solving convex nonlinear optimization problems

Let us consider the following general CNLP (Convex Nonlinear Programming) problem.

$$\begin{aligned} &\min\ f(x), \\ &\text{s.t}\ \ g(x) \leq 0, h(x) = 0. \end{aligned} \tag{CNLP}$$

where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, $g(x) = [g_1(x), g_2(x), ..., g_m(x)]^T$ is an m-dimensional vector-valued continuous function of n variables, and the functions $f, g_1, ..., g_m$ are assumed to be convex and twice differentiable. The constraint function $h(x) = Ax - b$, where $A \in \mathbb{R}^{(l\times n)}$ and $b \in \mathbb{R}^l$, represents a linear equality constraint.

The Lagrange function of (CNLP) is given by [101]

$$\mathcal{L}(x, u, v) = f(x) + \frac{1}{2}\sum_{k=1}^{m} u_k^2 g_k(x) + \sum_{p=1}^{l} v_p h_p(x). \tag{3.9}$$

where $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^l$ are the Lagrange multipliers. $x \in \mathbb{R}^n$ is an optimal solution of (CNLP) if and only if there exist $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^l$ such that the following Karush-Kuhn-Tucker (KKT) system is verified.

$$u \geq 0, g(x) \leq 0, u^T g(x) = 0, \tag{3.10}$$

$$\nabla f(x) + \nabla g(x)^T u + \nabla h(x)^T v = 0 \tag{3.11}$$

$$h(x) = 0. \tag{3.12}$$

Lemma 7. If f and g_k, $k = 1, ..., m$ are all convex, then x^* is an optimal solution of (CNLP) if and only if x^* is KKT point of (CNLP).

Now, consider the time-dependent variables $x(t)$, $u(t)$, and $v(t)$. Nazemi [101] designs the following neural network that settles at the saddle point of the Lagrangian function $\mathcal{L}(x, u, v)$.

$$\frac{dx}{dt} = -\nabla_x L(x, u, v), \tag{3.13}$$

$$\frac{du}{dt} = \nabla_u L(x, u, v), \tag{3.14}$$

$$\frac{dv}{dt} = \nabla_v L(x, u, v). \tag{3.15}$$

For the sake of simplicity the dynamical neural network (3.13)-(3.15) can be written as follows

$$\frac{dy}{dt} = \phi(y), \tag{3.16}$$

$$y(t_0) = y_0, u(t_0) \neq 0. \tag{3.17}$$

where $y = (x, u, v)^T$ and y_0 is a given initial points. Nazemi [101] proves the following results.

Theorem 8. Let $y* = (x^*, u^*, v^*)^T$ be the equilibrium of the neural network (3.16)-(3.17). Then, x^* is a KKT point of problem (CNLP).

Lemma 9. The equilibrium point of (3.16)-(3.17) is unique.

Lemma 10. For any initial point $y(t_0) = (x(t_0), u(t_0), v(t_0))^T$, there exists a unique continuous solution $y(t) = (x(t), u(t), v(t))^T$ for the system described by equations (3.16)-(3.17).

Theorem 11. If $\nabla f(x)$ is positive definite and $\nabla g_k(x)$, $k = 1, ..., m$ is positive semi-definite, or if $\nabla f(x)$ is positive semi-definite and $\nabla g_k(x)$, $k = 1, ..., m$ is positive definite, then the neural network (3.16)-(3.17) is globally stable in the Lyapunov sense. Additionally, it globally converges to a KKT point $y = (x, u, v)^T$, where x is the optimal solution of problem (CNLP), and u, v are the optimal solutions of its dual.

3.2 . Biconvex Optimization

3.2.1 . Introduction

Biconvex optimization refers to a class of optimization problems where the objective function is convex in each of its variables while keeping the other variables fixed. This type of optimization problem arises in various fields, including machine learning, signal processing, and information processing. Biconvex optimization models have been extensively studied due to their practical applications and mathematical properties. One popular approach for solving biconvex optimization problems is the alternating direction method of multipliers (ADMM), which is an iterative optimization algorithm that handles the separable structure of biconvex problems. ADMM has been widely used in biconvex optimization due to its efficiency and convergence guarantees.

In recent years, there have been several advancements and applications of biconvex optimization. In machine learning, biconvex optimization has been used for tasks such as matrix completion, collaborative filtering, and multi-view learning. In signal and information processing, biconvex optimization has been applied to problems like compressed sensing, image reconstruction, and channel estimation. Research in biconvex optimization has focused on developing efficient algorithms, understanding theoretical properties, and exploring applications in various domains. Different techniques, such as convex relaxation, non-convex relaxation, and convex-concave procedures, have been proposed to solve biconvex optimization problems. Additionally, research has investigated the impact of incorporating additional constraints, such as sparsity or low-rank structure, into biconvex optimization models. These extensions have led to improved performance and more accurate solutions in many practical scenarios.

3.2.2 . Definition and Properties

The study of convexity of functions is motivated by the property that if a function is convex, then every local minimum is guaranteed to be a global minimum. However, it is important to note that convexity is not a necessary condition for this local-global property.

Unfortunately, biconvexity, as defined, does not imply the local-global property. While biconvex functions possess certain desirable properties, they do not guarantee that all local minima will be global minima. Therefore, the study and understanding of biconvexity require additional considerations beyond the local-global property.

Definition 5. Let $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^m$ be two non-empty, convex sets and let $\mathcal{B} \subseteq \mathcal{X} \times \mathcal{Y}$. We define

$$\mathcal{B}_x = \{y \in \mathcal{Y} | (x, y) \in \mathcal{B}\} \tag{3.18}$$

and

$$\mathcal{B}_y = \{x \in \mathcal{X} | (x, y) \in \mathcal{B}\}. \tag{3.19}$$

The set $\mathcal{B}$ is said to be biconvex on $\mathcal{X} \times \mathcal{Y}$, if $\mathcal{B}_x$ is biconvex for every $x \in \mathcal{X}$ and $\mathcal{B}_y$ is biconvex for every $y \in \mathcal{Y}$.

Definition 6. A function $f : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}$ is called a *biconvex* function on $\mathbb{R}^n \times \mathbb{R}^m$, if

$$f_x(.) := f(x, .) : \mathbb{R}^m \longrightarrow \mathbb{R} \tag{3.20}$$

is convex on $\mathbb{R}^m$ for every fixed $x \in \mathbb{R}^n$ and

$$f_y(.) := f(., y) : \mathbb{R}^n \longrightarrow \mathbb{R} \tag{3.21}$$

is convex on $\mathbb{R}^n$ for every fixed $y \in \mathbb{R}^m$.

Theorem 12. [54] Let $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^m$, a function $f(x, y)$ is biconvex on $\mathcal{X} \times \mathcal{Y}$ if and only if for all quadruples $(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2) \in \mathcal{X} \times \mathcal{Y}$ and for every $\lambda, \beta \in [0, 1]$

$$f\left((1-\lambda)x_1 + \lambda x_2, (1-\beta)y_1 + \beta y_2\right) \leq (1-\lambda)(1-\beta)f(x_1, y_1) + \beta(1-\lambda)f(x_1, y_2) + \lambda(1-\beta)f(x_2, y_1) + \lambda\beta f(x_2, y_2)$$

Theorem 13. [54] If f is biconvex, then its level sets

$$\mathcal{L}_c = \{(x, y) \in \mathcal{X} \times \mathcal{Y} | f(x, y) \leq c\} \tag{3.22}$$

are all convex for every $c \in \mathbb{R}$.

Lemma 14. [57] Let f and g be two biconvex functions on $\mathcal{X} \times \mathcal{Y}$ and let $\mu \in \mathbb{R}^+$, then the functions $f + g$ and μf are biconvex.

Lemma 15. [57] Let f a biconvex function on $\mathcal{X} \times \mathcal{Y}$ and let $\phi : \mathbb{R} \longrightarrow \mathbb{R}$ a e a convex, non-decreasing function. Then the function $\phi(f(., .))$ is biconvex on $\mathcal{X} \times \mathcal{Y}$.

3.2.3 . Biconvex optimization problems

The existing research highlights the challenging nature of solving biconvex programming problems, making it a highly valuable area of study that necessitates the exploration of new theories and solution methods. In this section, we present two significant theoretical results related to the partial optimum of biconvex programming, utilizing the objective penalty function.

The first result establishes the equivalence between the partial Karush-Kuhn-Tucker (KKT) condition and the partial exactness property of the objective penalty function in biconvex programming. This finding provides an important insight into the relationship between the KKT condition and the behavior of the objective penalty function, enabling a deeper understanding of the optimization process.

The second result demonstrates the equivalence between the partial stability condition and the partial exactness property of the objective penalty function in biconvex programming. This result sheds light on the significance of the stability condition in relation to the objective penalty function, offering valuable insights into the convergence behavior of algorithms aiming to solve the partial optimum of biconvex programming.

These theoretical results offer crucial guarantees for the convergence of algorithms designed to tackle the partial optimum of biconvex programming problems. By establishing the equivalence between key conditions and properties of the objective penalty function, this research provides a solid foundation for developing efficient and effective algorithms that can reliably converge towards optimal solutions for biconvex programming.

We consider the following biconvex programming

$$\min f(x,y), \quad \text{subject to} \quad g_i(x,y) \leq 0, \;\; i = 1, ..., K, \tag{BP}$$

where f g_i, $i = 1, ..., K$ are biconvex functions on $\mathbb{R}^n \times \mathbb{R}^m$. the feasible set for (BP) is denoted by

$$\mathcal{B} = \{(x,y) \in \mathbb{R}^n \times \mathbb{R}^m \mid g_i(x,y) \leq 0, i = 1, ..., m\}.$$

We define

$$\mathcal{B}_x = \{y \in \mathbb{R}^m \mid g_i(x,y) \leq 0, i = 1, ..., m\} \text{ and } \mathcal{B}_y = \{x \in \mathbb{R}^n \mid g_i(x,y) \leq 0, i = 1, ..., m\}.$$

We additionally define two suboptimal problems

$$\min f(x,y), \quad \text{subject to} \quad x \in \mathcal{B}_y, \tag{BP1}$$

and

$$\min f(x,y), \quad \text{subject to} \quad y \in \mathcal{B}_x. \tag{BP2}$$

Definition 7. Let $(x^*, y^*) \in \mathcal{B}$, (x^*, y^*) is called a *partial optimum* of (BP) if

$$f(x^*, y^*) \leq f(x, y^*), \forall x \in \mathcal{B}_y \text{ and } f(x^*, y^*) \leq f(x^*, y), \forall y \in \mathcal{B}_x.$$

.

Remark 2. It is straightforward that an optimal solution for (BP) is a partial optimum of (BP).

Remark 3. In biconvex optimization problems, there always exists a partial optimal solution [69].

Definition 8. Let $(x^*, y^*) \in \mathcal{B}$, if there exist α_i, $i = 1, ..., K$ such that

$$\nabla f(x^*, y^*) + \sum_{i=1}^{K} \alpha_i \nabla g_i(x^*, y^*) = 0,$$

$$\alpha_i g_i(x^*, y^*) = 0, \alpha_i \geq 0, i = 1, ..., K,$$

then (x^*, y^*) is called a *KKT point* of (BP).

Definition 9. Let $(x^*, y^*) \in \mathcal{B}$, if there exist α_i^1, α_i^2, $i = 1, ..., K$ such that

$$\nabla_x f(x^*, y^*) + \sum_{i=1}^{K} \alpha_i^1 \nabla_x g_i(x^*, y^*) = 0,$$

$$\nabla_y f(x^*, y^*) + \sum_{i=1}^{K} \alpha_i^2 \nabla_y g_i(x^*, y^*) = 0,$$

$$\alpha_i^1 g_i(x^*, y^*) = 0, \alpha_i^2 g_i(x^*, y^*) = 0, \alpha_i^1 \geq 0, \alpha_i^2 \geq 0, i = 1, ..., K,$$

then (x^*, y^*) is called a *partial KKT point* of (BP).

Definition 10. Let $(x^*, y^*) \in \mathcal{B}$, if there exist $(\tilde{x}, \tilde{y})$ such that

$$g_i(x^*, \tilde{y}) < 0, g_i(\tilde{x}, y^*) < 0, i = 1, ..., K,$$

then (BP) is said to satisfy *partial Slater constraint* qualification at (x^*, y^*).

Theorem 16. [68] Let $(x^*, y^*) \in \mathcal{B}$. If (BP) satisfies partial Slater constraint qualification at (x^*, y^*), (x^*, y^*) is a partial optimum of (BP) if and only if (x^*, y^*) is a partial KKT point of (BP).

Corollary 17. [68] Let $(x^*, y^*) \in \mathcal{B}$ be a partial optimum of (BP). If (BP) satisfies partial Slater constraint qualification at (x^*, y^*), then $(x^*, y^*) \in \mathcal{B}$ is a KKT point of (BP) if and only if $\alpha_i^1 = \alpha_i^2$, $i = 1, ..., K$.

3.2.4 . Algorithms

In this subsection, we focus on various methods and algorithms specifically designed to solve biconvex minimization problems represented by (BP). These methods take advantage of the inherent biconvex structure of the problem to achieve efficient solutions. For each solution approach, we provide concise algorithmic descriptions and discuss their convergence properties and limitations.

Alternate convex search

Firstly, we explore the Alternate Convex Search (ACS) method, which is a special case of Block-Relaxation Methods. This approach involves iteratively optimizing each block of variables while keeping the other blocks fixed. By alternating between the blocks and iteratively refining the solution, this method aims to converge to a biconvex minimizer. We delve into the details of this algorithm, discuss its convergence properties, and highlight any limitations or challenges associated with its implementation. The pseudocode of the alternate convex search algorithm is given as follows.

Algorithm 1 The alternate convex search

1-Select an arbitrary starting point, denoted as $z_0 = (x_0, y_0) \in \mathcal{B}$.
2-Initialize the iteration counter $i = 0$.
3- For a fixed y_i, solve the convex problem (BP1).
if There exists an optimal solution $x^* \in \mathcal{B}_{y_i}$ for (BP1) **then**
  Set $x_{i+1} = x^*$,
else
  STOP.
end if
4- For a fixed x_i, solve the convex problem (BP2).
if There exists an optimal solution $y^* \in \mathcal{B}_{x_i}$ for (BP2) **then**
  Set $y_{i+1} = y^*$,
else
  STOP.
end if
5- Set $z_{i+1} = (x_{i+1}, y_{i+1})$.
if A stopping criterion is satisfied, i.e, $||z_{i+1} - z_i|| \leq \epsilon$ **then**
  STOP.
else
  $i = i + 1$.
end if

Theorem 18. [57] Assuming that $\mathcal{B} \subseteq \mathbb{R}^n \times \mathbb{R}^m$, where $f : \mathcal{B} \longrightarrow \mathbb{R}$ is bounded from below, and the optimization problems (BP1) and (BP2) are solvable, we can conclude that the sequence $\{f(z_i)\}$ generated Algorithm 1 converges monotonically.

The statement of Theorem 27 may appear weak because the boundedness of the objective function f guarantees the convergence of the sequence $\{f(z_i)\}$ generated by the ACS algorithm, but it does not automatically guarantee the convergence of the sequence $\{z_i\}$ itself. It is possible to encounter scenarios where the sequence $\{f(z_i)\}$ converges while the sequence $\{z_i\}$ diverges, as shown in Example 4.3 in [57].

In other words, while the objective function values approach a limit as the ACS algorithm iterates, the sequence of points $\{z_i\}$ may not converge to a specific point or region

in the optimization space. This behavior can occur even for biconvex functions, where the convergence of $\{f(z_i)\}$ does not imply the convergence of $\{z_i\}$.

The existence of such cases highlights the importance of considering additional conditions or assumptions to ensure the convergence of the sequence $\{z_i\}$. Simply relying on the boundedness of f may not be sufficient to guarantee the convergence of the optimization variables.

Therefore, it is crucial to carefully examine the specific properties of the optimization problem, such as additional constraints or assumptions, to ascertain the convergence of the sequence $\{z_i\}$.

Definition 11. [57] Let $\mathcal{B} \subseteq \mathbb{R}^n \times \mathbb{R}^m$, let $z_1 = (x_1, y_1)$ and $z_2 = (x_2, y_2)$ in $\mathcal{B}$. The map $A : \mathcal{B} \longrightarrow \mathcal{P}(\mathcal{B})$, where $\mathcal{P}(\mathcal{B})$ is the power set of $(\mathcal{B})$ defined by $z_2 \in A(z_1)$ if and only if

$$f(x_2, y_1) \leq f(x, y_1)\ \forall x \in \mathcal{B}_{y_1} \text{ and } f(x_2, y_2) \leq f(x_2, y)\ \forall y \in \mathcal{B}_{x_2}$$

is called *the algorithmic map* of the ACS algorithm.

In other words,$A(z_1) = z_2$ in Definition 21 implies that the ACS algorithm, based on the algorithmic map A, identifies the point z_2 as a candidate solution that satisfies the given conditions. This map plays a crucial role in guiding the ACS algorithm by determining the next point to explore in the optimization process.

By utilizing the algorithmic map A, the ACS algorithm dynamically selects points in $\mathcal{B}$, allowing for an iterative exploration of the optimization space and providing information about the neighborhood regions where local minima are identified, helping to refine the search and improve the convergence properties of the ACS algorithm.

Theorem 19. [57] Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be closed sets and let $f : X \times Y \longrightarrow \mathbb{R}$ be a continuous function. If the sequence $\{z_i\}$ generated by the ACS algorithm converges to $z^* \in X \times Y$, then z^* is a partial optimum of (BP).

To sum up, to seek the global optimum of a biconvex minimization problem using the ACS algorithm, a multistart version of ACS can be employed, as suggested by Goh et al. [54]. However, it is important to note that even with the multistart approach, there is no guarantee of finding the global optimum within a reasonable amount of time or being certain that the identified best minimum is indeed the global minimum. The multistart version of ACS involves running the ACS algorithm from multiple different starting points in order to explore different regions of the optimization space. By initiating the algorithm from various initial points, it increases the chances of discovering better solutions and potentially finding the global minimum. However, due to the inherent complexity of biconvex optimization problems, it is challenging to ensure global optimality. The presence of multiple local minima and the intricate interplay between the convex components make it difficult to guarantee that the identified solution is indeed the global minimum. Additionally, the computational time required to exhaustively search the entire optimization space can be prohibitive. Therefore, while the multistart version of ACS improves the likelihood of finding better solutions, it does not provide a foolproof guarantee of locating the

global optimum within a reasonable timeframe. It is crucial to carefully analyze the problem characteristics, employ suitable search strategies, and evaluate the obtained results to make informed decisions regarding the optimality of the solution and the trade-off between computational resources and solution quality. A detailed discussion with examples can be found in [57].

The global optimization algorithm

In this subsection, we present a comprehensive review of the Global Optimization Algorithm (GOP), developed by Floudas and Visweswaran [138], for solving constrained biconvex minimization problems. This algorithm leverages the convex substructure of the problem through a primal-relaxed dual approach, building upon decomposition concepts introduced by Benders [13] and Geoffrion [51].

The GOP algorithm follows a two-step process similar to the ACS method. In the first step, the constrained problem is solved for a fixed value of the variable y, leading to an upper bound on the solution of the biconvex problem. This initial problem is referred to as the primal problem.

To obtain a lower bound for the solution, the algorithm utilizes duality theory and applies linear relaxation. The resulting relaxed dual problem is solved for every possible combination of bounds in a subset of the variables x.

By iteratively alternating between the primal and relaxed dual problems, the GOP algorithm achieves finite ϵ-convergence to the global optimum. This means that as the iterations progress, the algorithm converges towards the global optimum within a specified tolerance ϵ.

The primal-relaxed dual approach of the GOP algorithm combines decomposition techniques, duality theory, and linear relaxation to effectively explore the solution space of constrained biconvex minimization problems. Through this iterative process, the algorithm provides a robust framework for finding globally optimal solutions.

The finite ϵ-convergence guarantees that the algorithm will eventually converge to the global optimum, although the convergence rate and the number of iterations required may vary depending on the specific problem instance and input parameters.

Theorem 20. [47] If the conditions of Theorem 4.11 [57] hold, then the GOP algorithm will terminate at the global optimum of the biconvex minimization problem.

Advantages of the GOP algorithm

- Convexity of the primal problem: The fact that the primal problem in the first step of each iteration is a convex problem is a major advantage. This means that every local optimum is also the global minimum of the subproblem. This property simplifies the optimization process and ensures that the obtained solution is globally optimal within the subproblem.

- Simplification of constraints: The set of constraints for the convex subproblem often simplifies to linear or quadratic constraints in the x-variables. This simplification

enables the use of conventional non-linear local optimization solvers to efficiently solve the primal problem.

- Reduction of variables: The relaxed dual problem only needs to be solved for the connected x-variables, reducing the number of variables involved. This reduction in variables can lead to improved computational efficiency and reduced computational resources.

Drawbacks of the GOP algorithm

- Number of subproblems: In each iteration of the algorithm, a potentially large number of non-linear subproblems have to be solved to obtain a new lower bound for the problem. This can be computationally expensive, especially when dealing with a large number of connected x-variables.
- Total enumeration: The algorithm requires a total enumeration of all possible assignments of the connected x-variables to their lower and upper bounds in each iteration. This exhaustive search can be time-consuming and computationally demanding, particularly for problems with a large search space.
- Potential complexity: Depending on the structure of the given biconvex problem, the number of relaxed dual problems may still be significant even with the improvements suggested in the literature. The complexity of solving these problems can impact the overall efficiency and scalability of the algorithm.

For a detailed survey on biconvex sets and optimization, we refer the reader to the work of Gorski, Pfeuffer and Klamroth [57].

4 - Joint Chance Constrained Geometric Optimization With Independent Row Vectors

In this chapter, we investigate a dynamical neural network approach to address non-convex geometric programs with joint probabilistic constraints. These programs involve coefficients that follow a normal distribution and independent matrix row vectors. We establish the stability and convergence of our neural network based on Lyapunov analysis. Additionally, we demonstrate the equivalence between the optimal solution of the deterministic equivalent problem of geometric programs with joint probabilistic constraints and the solution of the dynamical system. Finally, we present numerical experiments to showcase the performance of our approach compared to state-of-the-art methods.

4.1 . Geometric programs

This section is published in Mathematics and Computers in Simulation Journal [132].

We consider the following general form of geometric programs with joint probabilistic constraints

$$\begin{aligned} &\min_{t\in\mathbb{R}^M_{++}} \sum_{i\in I_0} c_i \prod_{j=1}^{M} t_j^{a_{ij}}, \\ &\text{s.t} \quad \mathbb{P}\left(\sum_{i\in I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \leq 1, k=1,....,K\right) \geq 1-\epsilon, \end{aligned} \tag{GPJC}$$

where $\{c_i\}_{i\in I_k}$, $k \in \{1,..,K\}$ are pairwise independent normally distributed random variables i.e $c_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, $i \in I_k$ where $\mu_i \geq 0$ is the mean value and σ_i is the standard deviation of c_i. The coefficients $a_{ij}, i \in I_k, j = 1,...,M$ are deterministic parameters, and $1-\epsilon$ is a given probability level with $\epsilon \in (0, 0.5]$.

4.1.1 . Deterministic equivalent problem

Using the pairwise independence between $\{c_i\}_{i\in I_k}$, the joint constraint of (GPJC) can be simplified to be equivalent to

$$\prod_{k=1}^{K} \mathbb{P}\left(\sum_{i\in I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \leq 1\right) \geq 1-\epsilon \tag{4.1}$$

Consequently, problem (GPJC) can be reformulated as follows [88]

$$\begin{aligned}
&\min_{t\in\mathbb{R}^M_{++}} \ \mathbb{E}\left[\sum_{i\in I_0} c_i \prod_{j=1}^{M} t_j^{a_{ij}}\right], \\
&\text{s.t} \ \ \mathbb{P}\left(\sum_{i\in I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \le 1\right) \ge y_k, \ k=1,...,K, \\
&\qquad \prod_{k=1}^{K} y_k \ge 1-\epsilon, \ 0\le y_k \le 1\,, k=1,...,K.
\end{aligned} \tag{4.2}$$

where y_k, $k=1,...,K$, are nonnegative auxiliary variables.
A deterministic equivalent problem of (4.2) is given by [25]

$$\begin{aligned}
&\min_{t\in\mathbb{R}^M_{++}} \ \sum_{i\in I_0} \mu_i \prod_{j=1}^{M} t_j^{a_{ij}}, \\
&\text{s.t} \ \ \sum_{i\in I_k} \mu_i \prod_{j=1}^{M} t_j^{a_{ij}} + \phi^{-1}(y_k)\sqrt{\sum_{i\in I_k} \sigma_i^2 \prod_{j=1}^{M} t_j^{a_{ij}}} \le 1, \ k=1,...,K, \\
&\qquad \prod_{k=1}^{K} y_k \ge 1-\epsilon, \ 0\le y_k \le 1\,, k=1,...,K,
\end{aligned} \tag{4.3}$$

where $\phi^{-1}(.)$ is the quantile of the standard normal distribution and can be represented using the inverse error function $\phi^{-1}(y_k) = \sqrt{2}\text{erf}^{-1}(2y_k-1)$.
We apply the log-transformation $r_j = \log(t_j),\ j=1,...,M$ and $x_k = \log(y_k)$, $k=1,...,K$, to (4.3) which leads to the following biconvex optimization problem.

$$\begin{aligned}
&\min \sum_{i\in I_0} \mu_i \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\}, \\
&\text{s.t} \ \ \sum_{i\in I_k} \mu_i \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\} \\
&\qquad + \sqrt{\sum_{i\in I_k} \sigma_i^2 \exp\left\{\sum_{j=1}^{M} (2a_{ij} r_j + \log(\phi^{-1}(e^{x_k})^2))\right\}} \le 1, \ k=1,...,K, \\
&\qquad \sum_{k=1}^{K} x_k \ge \log(1-\epsilon), \ x_k \le 0, \qquad\qquad k=1,...,K.
\end{aligned} \tag{4.4}$$

4.1.2 . Convex approximations

We introduce in this section two convex approximations proposed by Liu et al. [88] to solve (4.4). Later, to evaluate the performance of our proposed neurodynamic method we compare it with those approximations.

The expression $\log(\phi^{-1}(e^{x_k})^2)$ is approximated by the following piecewise linear function

$$F(x_k) = \max_{s=1,...,S} F_s(x_k), \tag{4.5}$$

where $F_s(x_k) \leq \log(\phi^{-1}(e^{x_k})^2)$, $\forall x_k \in [\log(1-\epsilon), 0)$ and $F_s(x_k) = d_s x_k + b_s$, $s = 1, ..., S$. One can choose the tangent lines of the expression $\log(\phi^{-1}(e^{x_k})^2)$ at various points within the interval $[\log(1-\epsilon), 0)$, denoted as $\zeta_1, \zeta_2, ..., \zeta_S$ and define

$$d_s = \frac{2e^{\zeta_s}(\phi^{-1})'(e^{\zeta_s})}{\phi^{-1}(e^{\zeta_s})}, \tag{4.6}$$

$$b_s = -d_s\zeta_s + \log(\phi^{-1}(e^{\zeta_s})^2). \tag{4.7}$$

We then obtain the following convex approximation of (4.4)

$$\begin{aligned}
&\min \sum_{i\in I_0} \mu_i \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\}, \\
&\text{s.t} \sum_{i\in I_k} \mu_i \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\} \\
&\quad + \sqrt{\sum_{i\in I_k} \sigma_i^2 \exp\left\{\sum_{j=1}^{M} (2a_{ij} r_j + d_s x_k + b_s)\right\}} \leq 1,\ s = 1, ..., S,\ k = 1, ..., K, \\
&\quad \sum_{k=1}^{K} x_k \geq \log(1-\epsilon),\ x_k \leq 0, \qquad\qquad k = 1, ..., K.
\end{aligned} \tag{4.8}$$

The optimal value of problem (4.8) represents a lower bound for problem (4.4). Additionally, as the number of tangent lines S approaches infinity, (4.8) converges to an equivalent reformulation of problem (4.4) [88].

To obtain an upper bound for the joint probabilistic constrained problem (GPJC), the widely used technique of sequential convex approximation was applied. The underlying principle of this approximation involves decomposing the original problem into subproblems, where a subset of variables is fixed alternately. In our specific problem, we begin by fixing $y = y^n$ and proceed to update t by solving the following subproblem

$$\begin{aligned}
&\min_{t\in\mathbb{R}^M_{++}} \sum_{i\in I_0} \mu_i \prod_{j=1}^{M} t_j^{a_{ij}}, \\
&\text{s.t} \sum_{i\in I_k} \mu_i \prod_{j=1}^{M} t_j^{a_{ij}} + \phi^{-1}(y_k^n) \sqrt{\sum_{i\in I_k} \sigma_i^2 \prod_{j=1}^{M} t_j^{a_{ij}}} \leq 1,\ k = 1, ..., K,
\end{aligned} \tag{4.9}$$

and then fix $t = t^n$, where t^n is the optimal solution of (4.9) and update y by solving the

subsequent subproblem

$$\begin{aligned} &\min_{y\in\mathbb{R}^K_+} \sum_{k=1}^{K} \psi_k y_k, \\ &\text{s.t} \quad y_k \leq \phi \left(\frac{1 - \sum\limits_{i\in I_k} \mu_i \prod\limits_{j=1}^{M} (t_j^n)^{a_{ij}}}{\sqrt{\sum\limits_{i\in I_k} \sigma_i^2 \prod\limits_{j=1}^{M} (t_j^n)^{a_{ij}}}} \right), \; k = 1, ..., K, \\ &\prod_{k=1}^{K} y_k \geq 1 - \epsilon, \; 0 \leq y_k \leq 1 \, , k = 1, ..., K, \end{aligned} \tag{4.10}$$

where ψ_k represents a selected searching direction, i.e, $\psi_k = \theta_k^n (\phi^{-1})'(y_k^n) \sqrt{\sum\limits_{i\in I_k} \sigma_i^2 \prod\limits_{j=1}^{M} (t_j^n)^{a_{ij}}}$ with t^n and θ^n represent an optimal solution and an optimal solution of the Lagrangian dual variable θ of problem (4.9), respectively.

4.1.3 . A dynamical neural network approach

Let $f(r) = \sum\limits_{i\in I_0} \mu_i \exp\left\{ \sum\limits_{j=1}^{M} a_{ij} r_j \right\}$, $h(r,x) = \log(1-\epsilon) - \sum\limits_{k=1}^{K} x_k$, $g_k(r,x) = x_k$ and $l_k(r,x) = \sum\limits_{i\in I_k} \mu_i \exp\left\{ \sum\limits_{j=1}^{M} a_{ij} r_j \right\} + \sqrt{\sum\limits_{i\in I_k} \sigma_i^2 \exp\left\{ \sum\limits_{j=1}^{M} (2a_{ij} r_j + \log(\phi^{-1}(e^{x_k})^2)) \right\}} - 1$. We can write problem (4.4) as follows:

$$\begin{aligned} &\min f(r), \\ &\text{s.t} \; l_k(r,x) \leq 0, \, k = 1, .., K, \\ &\quad h(r,x) \leq 0, \\ &\quad g_k(r,x) \leq 0, \, k = 1, .., K. \end{aligned} \tag{4.11}$$

The feasible set of (4.11) is given by

$$\mathsf{U} = \left\{ (r,x) \in \mathbb{R}^m \times \mathbb{R}^k \quad | \quad l_k(r,x) \leq 0, \quad h(r,x) \leq 0 \text{ and } g_k(r,x) \leq 0, \quad k = 1, .., K \right\}.$$

We define

$$\mathsf{U}(r) = \left\{ x \in \mathbb{R}^k \mid l_k(r,x) \leq 0, h(r,x) \leq 0 \text{ and } g_k(r,x) \leq 0, k = 1, .., K, \right\}$$

and

$$\mathsf{U}(x) = \{ r \in \mathbb{R}^m \mid l_k(r,x) \leq 0, h(r,x) \leq 0 \text{ and } g_k(r,x) \leq 0, k = 1, .., K \}$$

.

Let $(r^*, x^*) \in \mathbb{R}^m \times \mathbb{R}^K$ a partial KKT point of (4.11), then there exists $\lambda^{(r)}, \lambda^{(x)}, \beta_i^{(r)}, \beta_i^{(x)}$, $\gamma_i^{(r)}$ and $\gamma_i^{(x)}$, $i = 1, .., K$ such that

$$\nabla f(r^*) + \sum_{i=1}^{K} \beta_i^{(r)} \nabla_r l_i(r^*, x^*) + \lambda^{(r)} \nabla_r h(r^*, x^*) + \sum_{i=1}^{K} \gamma_i^{(r)} \nabla_r g_i(r^*, x^*) = 0, \quad (4.12)$$

$$\sum_{i=1}^{K} \beta_i^{(x)} \nabla_x l_i(r^*, x^*) + \lambda^{(x)} \nabla_x h(r^*, x^*) + \sum_{i=1}^{K} \gamma_i^{(x)} \nabla_x g_i(r^*, x^*) = 0, \quad (4.13)$$

$$\beta_i{}^{(r)} \geq 0, \quad \beta_i^{(r)} l_i(r^*, x^*) = 0, \quad \lambda^{(r)} \geq 0 \quad \lambda^{(r)} h(r^*, x^*) = 0, \quad i = 1, .., K, \quad (4.14)$$

$$\gamma_i^{(r)} \geq 0, \quad \gamma_i^{(r)} g_i(r^*, x^*) = 0, \quad \beta_i^{(x)} \geq 0, \quad \beta_i^{(x)} l_i(r^*, x^*) = 0, \quad i = 1, .., K, \quad (4.15)$$

$$\lambda^{(x)} \geq 0, \quad \lambda^{(x)} h(r^*, x^*) = 0, \quad \gamma_i^{(x)} \geq 0, \quad \gamma_i^{(x)} g_i(r^*, x^*) = 0, \quad i = 1, .., K, \quad (4.16)$$

Corollary 21. Let $(r^*, x^*) \in \mathbb{R}^M \times \mathbb{R}^K$ be a partial solution of (4.11), with respect to partial Slater constraints qualification at (r^*, x^*). Then (r^*, x^*) is a partial optimum of (4.11) if and only if the partial KKT system (4.12)-(4.16) holds. Furthermore, if $\lambda^{(r)} = \lambda^{(x)}$, $\beta^{(r)} = \beta^{(x)}$ and $\gamma^{(r)} = \gamma^{(x)}$, then (r^*, x^*) is a KKT point of (4.11).

Let $l(r, x) = (l_1(r, x), .., l_K(r, x))^T$ and $g(r, x) = (g_1(r, x), .., g_K(r, x))^T$, we propose a recurrent dynamical neural network model to solve problem (4.11). $r(.)$, $x(.)$, $\beta(.)$, $\lambda(.)$ and $\gamma(.)$ are time continuous variables. The dynamical neural network is driven by the following system.

$$\frac{dr}{dt} = -(\nabla f(r) + \nabla_r l(r, x)^T (\beta + l(r, x))_+ + \nabla_r h(r, x)^T (\lambda + h(r, x))_+ + \nabla_r g(r, x)^T (\gamma + g(r, x))_+), \quad (4.17)$$

$$\frac{dx}{dt} = -(\nabla_x l(r, x)^T (\beta + l(r, x))_+ + \nabla_x h(r, x)^T (\lambda + h(r, x))_+ + \nabla_x g(r, x)^T (\gamma + g(r, x))_+), \quad (4.18)$$

$$\frac{d\beta}{dt} = (\beta + l(r, x))_+ - \beta, \quad (4.19)$$

$$\frac{d\lambda}{dt} = (\lambda + h(r, x))_+ - \lambda, \quad (4.20)$$

$$\frac{d\gamma}{dt} = (\gamma + g(r, x))_+ - \gamma. \quad (4.21)$$

Let $z = (r, x, \beta, \lambda, \gamma)$, we can write the dynamical system (4.17)-(4.21) as follows

$$\begin{cases} \frac{dz}{dt} = \kappa \Phi(z), \\ z(t_0) = z_0, \end{cases} \quad (4.22)$$

where z_0 is the given initial point and κ is a scale parameter indicating the convergence rate of the neural network (4.17)-(4.21). For the sake of simplicity, we set $\kappa = 1$. Figure 4.12 illustrates the interconnections between the various inputs of the neural network (4.17)-(4.21).

In the following sections, we investigate the convergence and stability properties of the proposed dynamical neural network.

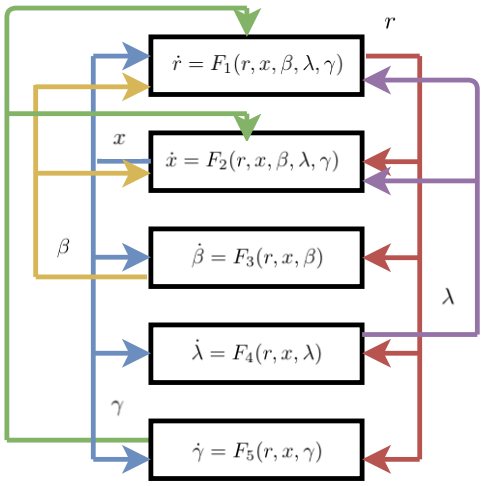

Figure 4.1: Feedback interconnection of neural network (4.17)-(4.21)

Theorem 22. Let $(r^*, x^*, \beta^*, \lambda^*, \gamma^*)$ an equilibrium point of the neural network defined by (4.17)-(4.21), then (r^*, x^*) is a KKT point of (4.11). On the other hand, if $(r^*, x^*) \in \mathbb{R}^M \times \mathbb{R}^K$ is a KKT point of (4.4), then there exists $\beta^* \geq 0$, $\lambda^* \geq 0$ and $\gamma^* \geq 0$ such that $(r^*, x^*, \beta^*, \lambda^*, \gamma^*)$ is an equilibrium point of the Neural Network (4.17)-(4.21).

Proof. Let $(r^*, x^*, \beta^*, \lambda^*, \gamma^*)$ an equilibrium point of the neural network defined by (4.17)-(4.21). Then, $\frac{dr^*}{dt} = 0$, $\frac{dx^*}{dt} = 0$, $\frac{d\beta^*}{dt} = 0$, $\frac{d\lambda^*}{dt} = 0$ and $\frac{d\gamma^*}{dt} = 0$. We have

$$-(\nabla f(r^*) + \nabla_r l(r^*, x^*)^T(\beta^* + l(r^*, x^*))_+ + \nabla_r h(r^*, x^*)^T(\lambda^* + h(r^*, x^*))_+ + \nabla_r g(r^*, x^*)^T(\gamma^* + g(r^*, x^*))_+) = 0,$$
$$-(\nabla_x l(r^*, x^*)^T(\beta^* + l(r^*, x^*))_+ + \nabla_x h(r^*, x^*)^T(\lambda^* + h(r^*, x^*))_+ + \nabla_x g(r^*, x^*)^T(\gamma^* + g(r^*, x^*))_+) = 0,$$
$$(\beta^* + l(r^*, x^*))_+ - \beta^* = 0,\ (\lambda^* + h(r^*, x^*))_+ - \lambda^* = 0,\ (\gamma^* + g(r^*, x^*))_+ - \gamma^* = 0.$$

Observe that $(\beta^* + l(r^*, x^*))_+ - \beta^* = 0$ if and only if $\beta^* \geq 0$, $l(r^*, x^*) \leq 0$ and ${\beta^*}^T l(r^*, x^*) = 0$. Similarly, we have $\lambda^* \geq 0$, $h(r^*, x^*) \leq 0$, ${\lambda^*}^T h(r^*, x^*) = 0$, $\gamma^* \geq 0$, $g(r^*, x^*) \leq 0$ and ${\gamma^*}^T g(r^*, x^*) = 0$.
By substitution in (4.17)-(4.21), we obtain the partial KKT system (4.12)-(4.16) with $\lambda^{(r)} = \lambda^{(x)} = \lambda$, $\beta^{(r)} = \beta^{(x)} = \beta$ and $\gamma^{(r)} = \gamma^{(x)} = \gamma$. The reverse implication of the theorem is straightforward. □

In order to establish the stability and convergence of the neural network, our first step is to demonstrate that the Jacobian matrix $\nabla\Phi(z)$ is a negative semidefinite matrix.

Lemma 23. The Jacobian matrix $\nabla\Phi(z)$ is negative semidefinite matrix.

Proof. We assume the existence of $0 < p, q < K$ such that
$(\beta + l)_+ = (\beta_1 + l_1(r, x), \beta_2 + l_2(r, x), ..., \beta_p + l_p(r, x), \underbrace{0, ..., 0}_{K-p})$,

$(\gamma + g)_+ = (\gamma_1 + g_1(r,x), \gamma_2 + g_2(r,x), ..., \gamma_q + g_q(r,x), \underbrace{0, ..., 0}_{K-q})$.

Assuming without loss of generality that $\lambda + h(r,x) \neq 0$, we represent the Jacobian matrix $\nabla\Phi$ as follows

$$\nabla\Phi(z) = \begin{bmatrix} A_1 & A_2 & A_3 & A_4 & A_5 \\ B_1 & B_2 & B_3 & B_4 & B_5 \\ C_1 & C_2 & C_3 & C_4 & C_5 \\ D_1 & D_2 & D_3 & D_4 & D_5 \\ E_1 & E_2 & E_3 & E_4 & E_5 \end{bmatrix},$$

where

$$A_1 = -(\nabla^2 f(r) + \sum_{i=1}^{p}((\beta_i + l_i)\nabla_r^2 l_i^p(r,x)) + \nabla_r l^p(r,x)^T \nabla_r l^p(r,x)) + (\lambda + h)\nabla_r^2 h(r,x)$$
$$+ \nabla_r h(r,x)^T \nabla_r h(r,x)) \quad + \sum_{i=1}^{q}((\gamma_i + g_i)\nabla_r^2 g_i^q(r,x)) \quad + \nabla_r g^q(r,x)^T \nabla_r g^q(r,x)),$$

$$A_2 = -(\sum_{i=1}^{p}((\beta_i + l_i)\nabla_x \nabla_r l_i^p(r,x)) + \nabla_r l^p(r,x)^T \nabla_x l^p(r,x)) + (\lambda + h)\nabla_x \nabla_r h(r,x)$$
$$+ \nabla_x h(r,x)^T \nabla_r h(r,x)) \quad + \sum_{i=1}^{q}((\gamma_i + g_i)\nabla_x \nabla_r g_i^q(r,x)) \quad + \nabla_r g^q(r,x)^T \nabla_x g^q(r,x)),$$

$$B_1 = -(\sum_{i=1}^{p}((\beta_i + l_i)\nabla_r \nabla_x l_i^p(r,x)) + \nabla_r l^p(r,x)^T \nabla_x l^p(r,x)) + (\lambda + h)\nabla_r \nabla_x h(r,x)$$
$$+ \nabla_x h(r,x)^T \nabla_r h(r,x)) \quad + \sum_{i=1}^{q}((\gamma_i + g_i)\nabla_r \nabla_x g_i^q(r,x)) \quad + \nabla_x g^q(r,x)^T \nabla_r g^q(r,x)),$$

$$B_2 = -(\sum_{i=1}^{p}(\beta_i + l_i)\nabla_x^2 l_i^p(r,x) + \nabla_x l^p(r,x)^T \nabla_x l^p(r,x)) + (\lambda + h)\nabla_x^2 h(r,x)$$
$$+ \nabla_x h(r,x)^T \nabla_x h(r,x) + \sum_{i=1}^{q}((\gamma_i + g_i)\nabla_x^2 g_i^q(r,x)) + \nabla_x g^q(r,x)^T \nabla_x g^q(r,x)),$$

$$A_3 = -C_1 = -\nabla_r l^p(r,x)^T, \quad A_4 = -D_1 = -\nabla_r h(r,x)^T, \quad A_5 = -E_1 = -\nabla_r g^q(r,x)^T,$$
$$B_3 = -C_2 = -\nabla_x l^p(r,x)^T, \quad B_4 = -D_2 = -\nabla_x h(r,x)^T, \quad B_5 = -E_2 = -\nabla_x g^q(r,x)^T,$$

$$C_3 = -S_p = \begin{bmatrix} O_{p\times p} & O_{p\times(K-p)} \\ O_{(K-p)\times p} & I_{(K-p)\times(K-p)} \end{bmatrix}, \qquad E_5 = -S_q = \begin{bmatrix} O_{q\times q} & O_{q\times(K-q)} \\ O_{(K-q)\times q} & I_{(K-q)\times(K-q)} \end{bmatrix},$$

$C_4 = 0$, $C_5 = 0$, $D_3 = 0$, $D_4 = 0$, $D_5 = 0$, $E_3 = 0$ and $E_4 = 0$.
Since g, h and l are twice differentiable, by Schwarz's theorem, we have $\nabla_r \nabla_x g_i^q(x,y) =$

$\nabla_x \nabla_r g_i^q(x,y), \forall i \in \{1,..,q\}$, $\nabla_r \nabla_x l_i^p(x,y) = \nabla_x \nabla_r l_i^p(x,y), \forall i \in \{1,..,p\}$ and $\nabla_r \nabla_x h(x,y) = \nabla_x \nabla_r h(x,y)$. It follows that $A_2 = B_1^T$ and $\nabla\Phi(z)$ becomes

$$\nabla\Phi(z) = \begin{bmatrix} A_1 & B_1^T & A_3 & A_4 & A_5 \\ B_1 & B_2 & B_3 & B_4 & B_5 \\ -A_3 & -B_3 & -S_p & 0 & 0 \\ -A_4 & -B_4 & 0 & 0 & 0 \\ -A_5 & -B_5 & 0 & 0 & -S_q \end{bmatrix}.$$

Demonstrating that $-S_p$ and $-S_q$ are negative semidefinite is straightforward. Hence, we can conclude that $S = \begin{bmatrix} -S_p & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -S_q \end{bmatrix}$ is negative semidefinite.
Given that the function f is convex and twice differentiable, we can conclude that the Hessian matrix $\nabla^2 f(x)$ is positive semidefinite. Furthermore, since g, h, and l are biconvex and twice differentiable, it follows from [57] that the Hessian matrices $\nabla_x^2 g_i^q(x,y)$, $\nabla_y^2 g_i^q(x,y)$, $\nabla_x^2 h(x,y)$, $\nabla_y^2 h(x,y)$, $\nabla_x^2 l_i^p(x,y)$, and $\nabla_y^2 l_i^p(x,y)$ are positive semidefinite for $i = 1, \ldots, q$. Consequently, we can conclude that A_1 and B_2 are negative semidefinite and then $A = \begin{bmatrix} A_1 & B_1^T \\ B_1 & B_2 \end{bmatrix}$ is negative semidefinite.
Let $B = \begin{bmatrix} A_3 & A_4 & A_5 \\ B_3 & B_4 & B_5 \end{bmatrix}$, $\nabla\Phi(z)$ can be written as $\nabla\Phi(z) = \begin{bmatrix} A & B \\ -B^T & S \end{bmatrix}$. Since S and A are negative semidefinite, the conclusion follows [48]. The proof when $\lambda + h(r,x) = 0$ follows along the same lines. □

Before proving the stability of our neural network, we introduce the following definition and lemma.

Definition 12. [117] A mapping $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is said to be monotonic if

$$(x-y)^T(F(x) - F(y)) \geq 0, \qquad \forall x, y \in \mathbb{R}^n.$$

Lemma 24. [117]. A differentiable mapping $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is monotonic, if and only if the jacobian matrix $\nabla F(x)$, $\forall x \in \mathbb{R}^n$, is positive semidefinite.

Theorem 25. The neural network (4.17)-(4.21) is stable in the Lyapunov sense and converges to $(r^*, x^*, \beta^*, \lambda^*, \gamma^*)$, where (r^*, x^*) is a KKT point of problem (4.11).

Proof. The proof of this theorem follows a similar approach to that of Theorem 3 in [141]. Consider $y^* = (r^*, x^*, \beta^*, \lambda^*, \gamma^*)$ as an equilibrium point for the neural network described by equations (4.17)-(4.21). We define the following Lyapunov function

$$V(y) = ||\Phi(y)||^2 + \frac{1}{2}||y - y^*||^2. \tag{4.23}$$

We have that $\frac{dV(y(t))}{dt} = (\frac{d\Phi}{dt})^T\Phi + \Phi^T\frac{d\Phi}{dt} + (y-y^*)^T\frac{dy}{dt}$ and since $\frac{d\Phi}{dt} = \frac{\nabla\Phi}{\nabla y}\frac{dy}{dt} = \nabla\Phi(y)\Phi(y)$, then $\frac{dV(y(t))}{dt} = \Phi^T(\nabla\Phi(y)^T + \nabla\Phi(y))\Phi + (y-y^*)^T\Phi(y)$. By Lemma 23, we have $\Phi^T(\nabla\Phi(y))\Phi \leq$

0 and $\Phi^T(\nabla\Phi(y)^T)\Phi \leq 0$.
By Lemma 23 and Lemma 24, we observe that $(y-y)^T(\Phi(y)) = (y-y)^T(\Phi(y)-\Phi(y)) \leq 0$. Hence, we can conclude that $\frac{dV(y(t))}{dt} \leq 0$.

Consequently, the neural network defined by equations (4.17)-(4.21) is stable at the equilibrium point y in the sense of Lyapunov. Since $V(y) \geq \frac{1}{2}\|y-y^*\|^2$, there exists a convergent subsequence $(y(t_k)_{k\geq 0})$ such that $\lim_{k\longrightarrow\infty} y(t_k) = \hat{y}$ and $\frac{dV(\hat{y}(t))}{dt} = 0$. Let $M = \{y(t) | \frac{dV(y(t))}{dt} = 0\}$, using LaSalle's invariance principle [127] we can see that any solution starting from a certain y_0 converges to the largest invariant set contained in M.
Notice that $\begin{cases} \frac{dr}{dt} = 0 \\ \frac{dx}{dt} = 0 \\ \frac{d\beta}{dt} = 0 \\ \frac{d\lambda}{dt} = 0 \\ \frac{d\gamma}{dt} = 0 \end{cases} \Leftrightarrow \frac{dV(y)}{dt} = 0$. It follows that $\hat{y}$ is an equilibrium point of the neural network (4.17)-(4.21).
Now we consider a new Lyapunov function defined by $\hat{V}(y) = \|\Phi(y)\|^2 + \frac{1}{2}\|y-\hat{y}\|^2$. Since $\hat{V}$ is continuously differentiable, $\hat{V}(\hat{y}) = 0$ and $\lim_{k\longrightarrow\infty} y(t_k) = \hat{y}$ then $\lim_{t\longrightarrow\infty}\hat{V}(y(t)) = \hat{V}(\hat{y}) = 0$. Furthermore, since $\frac{1}{2}\|y-\hat{y}\|^2 \leq \hat{V}(y(t))$ then $\lim_{t\longrightarrow\infty}\|y-\hat{y}\| = 0$ and $\lim_{t\longrightarrow\infty} y(t) = \hat{y}$. We conclude that the neural network (4.17)-(4.21) is convergent in the sense of Lyupanov to an equilibrium point $\hat{y} = (\hat{r}, \hat{x}, \hat{\beta}, \hat{\lambda}, \hat{\gamma})$ where $(\hat{r}, \hat{x})$ is a KKT point of problem (4.11). □

4.1.4 . Numerical experiments

To assess the performance of our approach, we investigate a transportation problem that involves optimizing the shape of a transportation box while adhering to geometric constraints. Initially, we focus on a three-dimensional scenario and subsequently extend it to larger instances to showcase the effectiveness of our approach.
For implementation, we utilize Python to develop our dynamical neural network. We employ numpy.random to randomly generate problem instances and use solve_ivp from the scipy.integrate package to solve the system of ordinary differential equations (ODEs). The deterministic equivalent programs are solved using the gekko package, while autograd.grad and autograd.jacobian are employed to compute gradients and partial derivatives.
To evaluate the performance, we compare our neural network approach with a state-of-the-art convex approximation method proposed by Liu et al. [88] presented in Section 4.1.2. Our assessment primarily focuses on the quality of the solution, and we do not record the CPU time as the current ODE solvers tend to be time-consuming.
All experiments are conducted on an Intel(R) Core(TM) i7-10610U CPU @ 1.80 GHz.

4.1.5 . A three-dimension shape optimization problem

We first consider a transportation problem involving the transfer of grain from a warehouse to a factory. The grain is transported using an open rectangular box with dimensions of length x_1 meters, width x_2 meters, and height x_3 meters, as illustrated in Figure 4.2. The primary objective is to maximize the volume of the rectangular box, which is given

by $x_1x_2x_3$. There are two additional constraints imposed on the floor area A_{floor} and the wall area A_{wall} of the box to ensure it conforms to the specified shape of a given truck. To account for the stochastic nature of the problem, we assume that the wall area A_{wall} and the floor area A_{floor} are independent random variables. The minimization problem

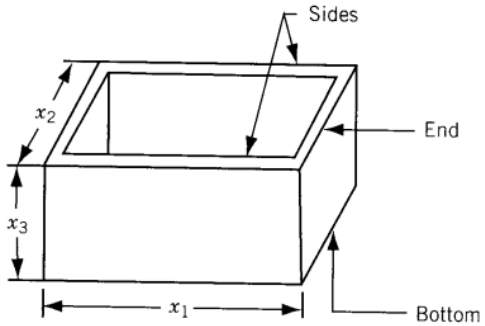

Figure 4.2: 3D-box shape [1]

can be formulated as follows

$$\begin{aligned}
\min_{x\in\mathbb{R}^3_{++}} \quad & x_1^{-1}x_2^{-1}x_3^{-1}, \\
\text{s.t} \quad & \mathbb{P}\left(\frac{1}{A_{wall}}(2x_3x_2+2x_1x_3)\leq 1, \frac{1}{A_{floor}}x_1x_2\leq 1\right)\geq 1-\epsilon. \\
& \alpha x_1^{-1}x_2\leq 1, \\
& \beta x_2^{-1}x_2\leq 1, \\
& \gamma x_2^{-1}x_3\leq 1, \\
& \delta x_3^{-1}x_2\leq 1,
\end{aligned} \tag{4.24}$$

The deterministic equivalent problem of (4.24) can be written as

$$\begin{aligned}
\text{minimize}_{\,x\in\mathbb{R}^3_{++}} \quad & x_1^{-1}x_2^{-1}x_3^{-1}, \\
\text{s.t} \quad & \mathbb{P}\left(\frac{1}{A_{wall}}(2x_3x_2+2x_1x_3)\leq 1\right)\geq y_1, \\
& \mathbb{P}\left(\frac{1}{A_{floor}}x_1x_2\leq 1\right)\geq y_2, \\
& \alpha x_1^{-1}x_2\leq 1, \\
& \beta x_2^{-1}x_2\leq 1, \\
& \gamma x_2^{-1}x_3\leq 1, \\
& \delta x_3^{-1}x_2\leq 1, \\
& y_1y_2\geq 1-\epsilon, \\
& 0\leq y_1,y_2\leq 1.
\end{aligned} \tag{4.25}$$

Deterministic Approach		Individual constraints		Joint constraints	
Obj value	VS	Obj value	VS	Obj value	VS
0.125	100	0.131	4	0.132	0

Table 4.1: Comparison of our neural network vs the deterministic formulation for the 3D-transportation problem

Regarding the numerical experiments, we have set the values as follows: $\epsilon = 0.1$, $\frac{1}{A_{\text{wall}}} \sim \mathcal{N}(0.05, 0.01)$, $\frac{1}{A_{\text{floor}}} \sim \mathcal{N}(0.5, 0.01)$, $\alpha = \beta = \gamma = \delta = 0.5$.
In addition to solving the stochastic problem (4.24), we solve also the corresponding deterministic problem where the random variables $\frac{1}{A_{\text{wall}}}$ and $\frac{1}{A_{\text{floor}}}$ are replaced by their means. We compare the results of the individual and joint probabilistic constraints formulations.
To evaluate the robustness of the deterministic formulation compared to the individual and joint probabilistic formulations, we have randomly generated a set of 100 instances of the stochastic variables $\frac{1}{A_{\text{wall}}}$ and $\frac{1}{A_{\text{floor}}}$ according to the mentioned means and variances. We have checked whether the solutions of the three formulations are feasible for problem (6.33) for each of the 100 scenarios. If the solutions are not feasible for a scenario, we label it as a violated scenario (VS).
The numerical results are summarized in Table 4.1. The first column represents the objective value obtained by the deterministic approach. The second column shows the number of violated scenarios out of the 100 scenarios. Columns three to six present the numerical results for the individual and joint chance-constrained formulations, respectively. The table indicates that our neural network successfully solves both the individual and joint chance-constrained 3D-transportation problems. It is worth noting that the joint formulation exhibits the highest level of robustness, as none of the 100 scenarios are violated, whereas the deterministic approach violates all 100 scenarios, as depicted in Figure 4.3.

In order to assess the performance of our approach, we compare our solution with the sequential convex approximation algorithm presented in Section 4.1.2. The numerical results are provided in Table 4.2. The first and second columns represent the objective value and the number of violated scenarios obtained using our neural network, respectively. The third and fourth columns display the objective value and the number of violated scenarios obtained using the sequential algorithm, respectively. The fifth column presents the gap between the two objective values, denoted as GAP, which is calculated as $\frac{(\text{Obj valueSA} - \text{Obj valueNN})}{\text{Obj valueSA}} \times 100$, where Obj valueNN and Obj value_{SA} represent the objective values obtained with the neural network and the sequential algorithm, respectively.
Table 4.2 demonstrates that the upper bounds obtained by our neural network and the sequential algorithm are identical. However, the sequential algorithm violates 4 scenarios, while our neural network solution remains feasible for all scenarios, as depicted in Figure 4.4.

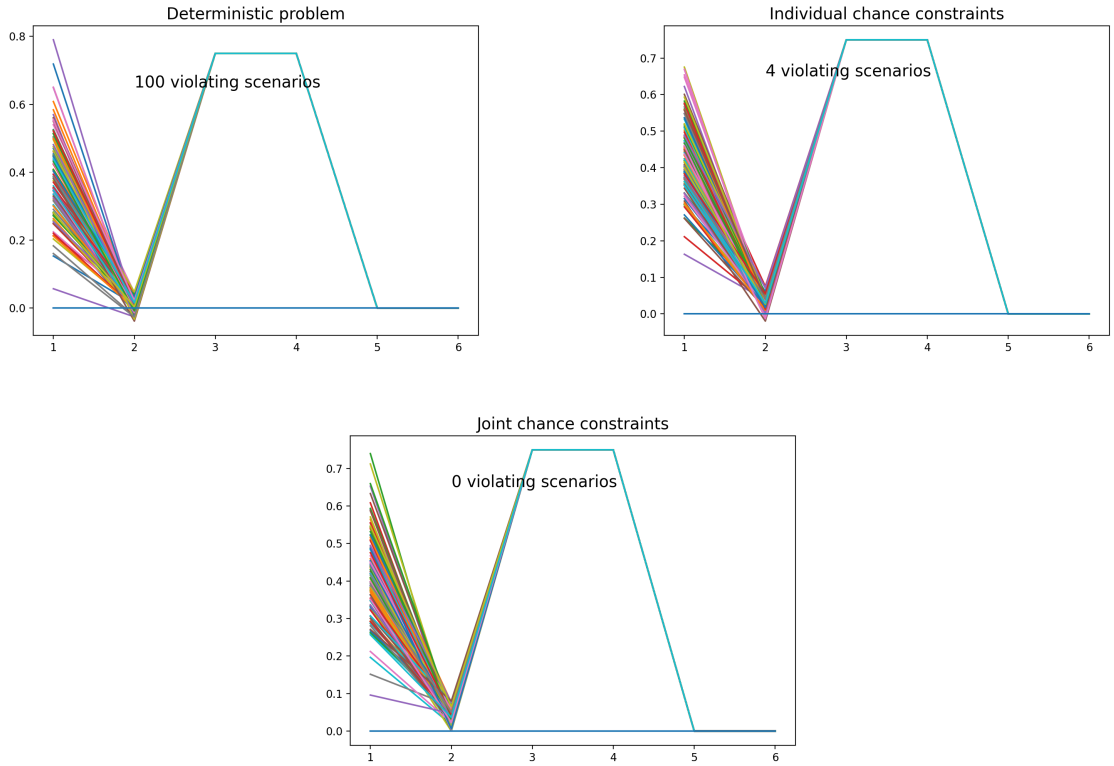

Figure 4.3: Number of VS for the deterministic formulation, the individual and joint chance constraints

Neural Network		Sequential Algorithm		
Obj value	VS	Obj value	VS	GAP
0.132	0	0.132	4	0.0

Table 4.2: Comparison between our neural network and the sequential algorithm

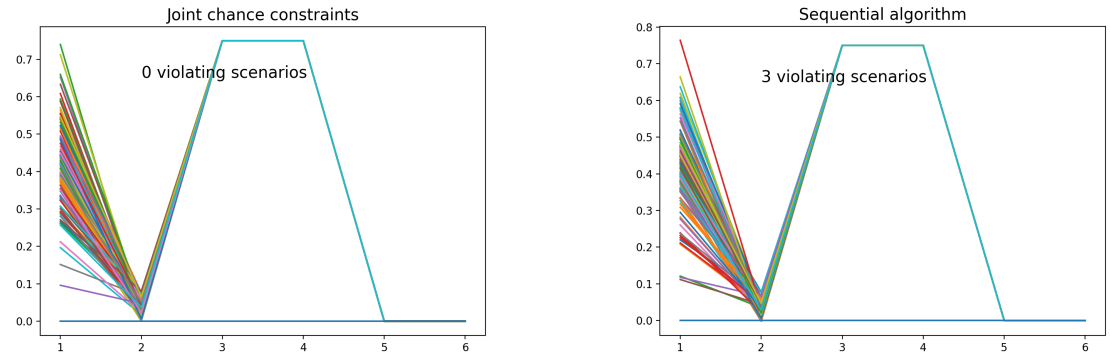

Figure 4.4: Number of VS of our neural network and the sequential algorithm

4.1.6 . Multidimensional transportation problem

In order to evaluate the performance of our approach on large-size instances, we extend the transportation problem studied in the previous subsection to higher-dimensional problems. The problem formulation for the m-dimensional case can be written as follows.

$$\begin{aligned} \min_{x\in\mathbb{R}^M_{++}} \ & \prod_{i=1}^{m} x_i^{-1}, \\ \text{s.t} \quad & \mathbb{P}\Big(\sum_{j=1}^{m-1}\Big(\frac{m-1}{A_{wall\,j}}x_1 \prod_{i=1,i\neq j}^{m} x_i\Big) \leq 1, \frac{1}{A_{floor}}\prod_{j=2}^{m} x_j \leq 1, \\ & \frac{1}{\gamma_{i,j}} x_i x_j^{-1} \leq 1, 1 \leq i \neq j \leq m\Big) \geq 1-\epsilon. \end{aligned} \tag{4.26}$$

For the numerical experiments, we set the following parameters: $\epsilon = 0.05$, $\frac{1}{A_{floor}}$ follows a normal distribution with a mean of $1.0/20.0$ and a standard deviation of 0.1, $\frac{1}{A_{wall_j}}$ is drawn from a normal distribution with a mean randomly selected from the interval $[1.0/60.0, 1.0/40.0]$ and a standard deviation randomly selected from the interval $[0.3, 0.5]$, and $\frac{1}{\gamma_{i,j}}$ follows a normal distribution with a mean of 0.5 and a standard deviation of 0.1. The results for different values of m for problem (4.26) are given in Table 4.4 where column one gives the data of the problem, e.g., (2, 4) means the problem is composed of two variables and four constraints. Columns two and three show the optimal value obtained by the deterministic approach and the number of VS over 100 generated scenarios, respectively. Columns four to six give the objective value of the individual chance constrained problem, the number of VS and the gap with the deterministic problem, respectively. We note that the gap is computed compared to the deterministic approach. The last three columns give the optimal value obtained by the joint chance-constrained problem, the number of VS and the gap with the deterministic approach, respectively.
Our joint chance-constrained neural network demonstrates greater robustness compared to the individual chance constraint and the deterministic models. On average, our joint chance constraint neural network has 10 violated scenarios (VS), while the average numbers of VS for the individual chance constraint and the deterministic approach are 40 and 92, respectively. Figure 4.5 illustrates the distributions of VS for the three formulations. Please note that we have not recorded the CPU time as our neural network problems are solved using standard ODE solvers, which are currently more time-consuming compared to standard optimization methods.

In a similar way as for the previous subsection, we compare the results obtained by the neural network for solving (4.26) with those obtained using the sequential algorithm. The results are presented in Table 4.4, where column one indicates the number of variables, column two indicates the number of constraints, column three shows the objective value obtained using the neural network, column four shows the corresponding number of violated scenarios (VS), column five shows the objective value obtained by the sequential

Data	Deterministic Approach		Individual chance constraints			Joint chance constraints		
	Obj value	VS	Obj value	VS	GAP	Obj value	VS	GAP
(2,4)	0.020	63	0.026	17	23	0.028	10	30
(3,8)	0.062	88	0.080	24	29	0.088	11	41
(4,14)	0.111	97	0.130	38	28	0.144	5	41
(5,22)	0.209	100	0.254	54	21	0.273	4	30
(6,32)	0.355	99	0.400	45	12	0.421	13	18
(7,44)	0.520	100	0.607	59	16	0.648	12	24

Table 4.3: Comparison between the deterministic approach and the individual and joint chance constraint neural networks

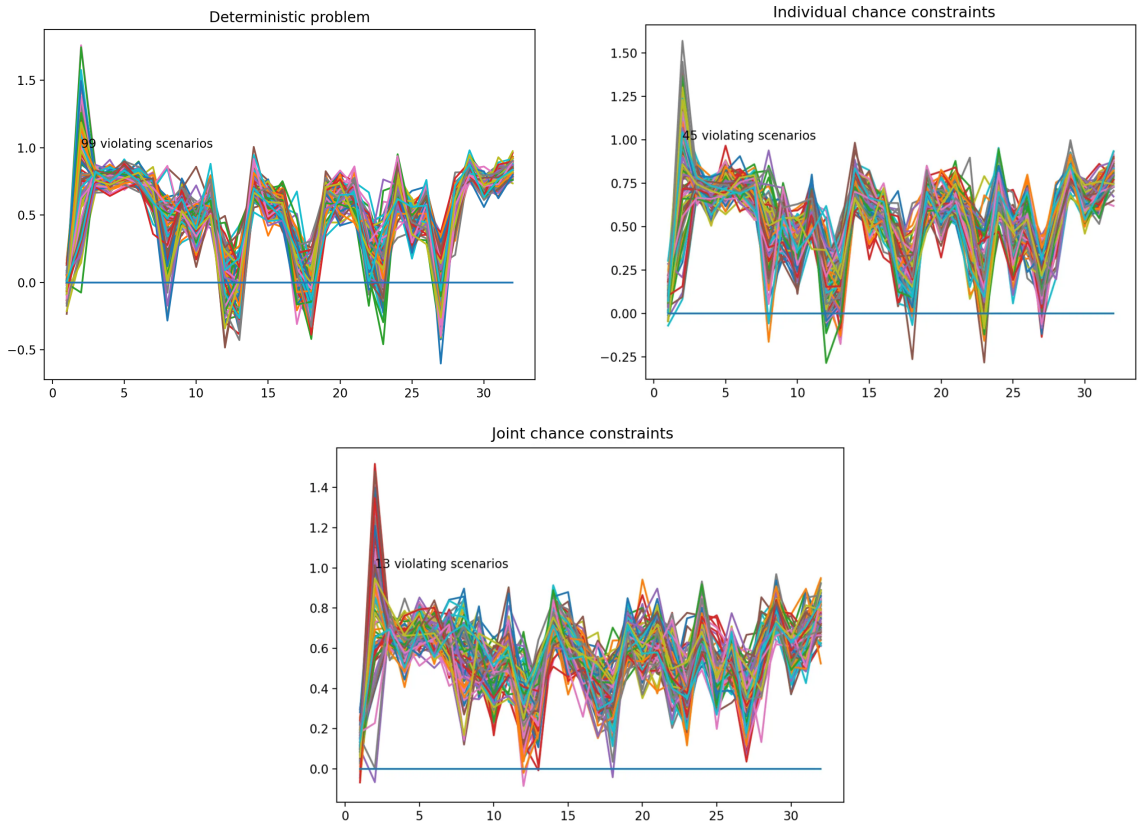

Figure 4.5: Number of VS for the deterministic formulation, the individual and joint chance constraints for 6 variables and 32 constraints

Var.Num	Const.Num	Neural Network	VS	Sequential Algorithm	VS	GAP
2	4	0.028	4	0.033	7	15.5
3	8	0.060	5	0.073	5	17.8
4	14	0.144	3	0.162	5	11.1
5	22	0.243	2	0.267	3	8.9
6	32	0.582	4	0.684	3	14.9

Table 4.4: Neural network vs. the sequential algorithm

algorithm, column six shows the corresponding number of VS, and column seven shows the gap between the two objective values. The gaps range from 8.9% up to 17.8%. Notice that the number of VS is comparable for the two approaches.

4.2 . Rectangular geometric programs

In this Section, we study the following stochastic rectangular programming problem

$$\min_{t\in\mathbb{R}^M_{++}} \mathbb{E}\left[\sum_{i\in I_0} c_i \prod_{j=1}^{M} t_j^{a_{ij}}\right], \tag{4.27}$$

$$\text{s.t.} \quad \mathbb{P}\left(\alpha_k \le \sum_{i\in I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \le \beta_k, k = 1, ..., K\right) \ge 1-\epsilon, \tag{4.28}$$

where $t = (t_j)_{1\le j\le M}$ is the decision variable, $c_i, i \in I_k$, $k = 0, 1, ..., K$ are uncorrelated normally distributed random variables , i.e., $c_i \sim \mathcal{N}(\bar{c}_i, \sigma_i^2)$, $\bar{c}_i \ge 0$. The coefficients $a_{ij}, i \in I_k, j = 1, ..., M$ are deterministic, $0 < \alpha_k < \beta_k$, and $1-\epsilon$ is a given probability level. Liu et al.[87] propose convex approximations based on the variable transformation to solve problem (4.27)-(4.28) with an elliptical distribution. They give upper and lower bounds for the optimal solution.

4.2.1 . Deterministic biconvex equivalent problem

Problem (4.27)-(4.28) is a joint constrained program. To transform the joint constraints into deterministic ones, we assume that the row vector constraints are mutually independent. Then, we introduce auxiliary variables y_k, $k = 1, ..., K$ and we rewrite the constraint (4.28) equivalently as

$$\mathbb{P}\left(\alpha_k \le \sum_{i\in I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \le \beta_k\right) \ge y_k, \, k = 1, ..., K, \tag{4.29}$$

$$\prod_{k=1}^{K} y_k \ge 1-\epsilon, 0 \le y_k \le 1, \, k = 1, ..., K. \tag{4.30}$$

The rectangular constraints (6.76) are equivalent to

$$\mathbb{P}\left(\sum_{i\in I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \geq \alpha_k\right) + \mathbb{P}\left(\sum_{i\in I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \leq \beta_k\right) - 1 \geq y_k, k = 1, ..., K. \tag{4.31}$$

Then, we introduce two additional K-dimensional auxiliary variables $z^+, z^- \in \mathbb{R}_+^K$ [87] such that (4.31) is equivalent to

$$\mathbb{P}\left(\sum_{i\in I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \geq \alpha_k\right) \geq z_k^+, k = 1, ..., K, \tag{4.32}$$

$$\mathbb{P}\left(\sum_{i\in I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \leq \beta_k\right) \geq z_k^-, k = 1, ..., K, \tag{4.33}$$

$$z_k^+ + z_k^- - 1 \geq y_k, 0 \leq z_k^-; z_k^+ \leq 1, k = 1, ..., K. \tag{4.34}$$

Deterministic reformulations of constraints (4.32) and (4.33) are given as follows [25]

$$-\sum_{i\in I_k} \bar{c}_i \prod_{j=1}^{M} t_j^{a_{ij}} + \phi^{-1}(z_k^+)\sqrt{\sum_{i\in I_k} \sigma_i^2 \prod_{j=1}^{M} t_j^{2a_{ij}}} \leq -\alpha_k, k = 1, ..., K, \tag{4.35}$$

$$\sum_{i\in I_k} \bar{c}_i \prod_{j=1}^{M} t_j^{a_{ij}} + \phi^{-1}(z_k^-)\sqrt{\sum_{i\in I_k} \sigma_i^2 \prod_{j=1}^{M} t_j^{2a_{ij}}} \leq \beta_k, k = 1, ..., K. \tag{4.36}$$

A biconvex equivalent formulation of constraint (4.79) can be obtained using a simple logarithmic transformation. Constraint (4.35) can be reformulated as follows

$$\phi^{-1}(z_k^+)^2 \sum_{i\in I_k} \sigma_i^2 \prod_{j=1}^{M} t_j^{2a_{ij}} \leq \left(\sum_{i\in I_k} \bar{c}_i \prod_{j=1}^{M} t_j^{a_{ij}} - \alpha_k\right)^2, k = 1, ..., K. \tag{4.37}$$

We write (4.37) equivalently for a given k as

$$2\alpha_k \sum_{i\in I_k} \bar{c}_i \prod_{j=1}^{M} t_j^{a_{ij}} - \sum_{i\in I_k}\sum_{p\in I_k} \bar{c}_i\bar{c}_p \prod_{j=1}^{M} t_j^{(a_{ij}+a_{pj})} + \phi^{-1}(z_k^+)^2 \sum_{i\in I_k} \sigma_i^2 \prod_{j=1}^{M} t_j^{2a_{ij}} \leq \alpha_k^2,$$

which can be reformulated as

$$\frac{2\alpha_k \sum\limits_{i\in I_k} \bar{c}_i \prod\limits_{j=1}^{M} t_j^{a_{ij}} + \phi^{-1}(z_k^+)^2 \sum\limits_{i\in I_k} \sigma_i^2 \prod\limits_{j=1}^{M} t_j^{2a_{ij}}}{\sum\limits_{i\in I_k}\sum\limits_{p\in I_k} \bar{c}_i\bar{c}_p \prod\limits_{j=1}^{M} t_j^{(a_{ij}+a_{pj})} + \alpha_k^2} \leq 1. \tag{4.38}$$

We approximate the denominator of constraint (4.38) with a monomial function by applying the arithmetic-geometric mean inequality [4].

$$\sum_{i\in I_k}\sum_{p\in I_k}\bar{c}_i\bar{c}_p\prod_{j=1}^{M}t_j^{(a_{ij}+a_{pj})}+\alpha_k^2\geq\prod_{i\in I_k}\prod_{p\in I_k}\left(\frac{\bar{c}_i\bar{c}_p\prod_{j=1}^{M}t_j^{(a_{ij}+a_{pj})}}{\delta_{ip}}\right)^{\delta_{ip}}\left(\frac{\alpha_k^2}{\delta_0}\right)^{\delta_0}, \tag{4.39}$$

where δ_0 and δ_{ip} are nonnegative parameter $\forall i\in I_k$ and $\delta_0+\sum_{i,p\in I_k}\delta_{ip}=1$.
Then, we approximate the denominator of constraint (4.38) with $\prod_{i\in I_k}\prod_{p\in I_k}\left(\frac{\bar{c}_i\bar{c}_p\prod_{j=1}^{M}t_j^{(a_{ij}+a_{pj})}}{\delta_{ip}}\right)^{\delta_{ip}}\left(\frac{\alpha_k^2}{\delta_0}\right)^{\delta_0}$. We write consequently problem (2.1)-(2.36) as

$$\min_{t\in\mathbb{R}_{++}^M,y,z^+,z^-}\ \sum_{i\in I_0}\bar{c}_i\prod_{j=1}^{M}t_j^{a_{ij}}, \tag{4.40}$$

$$\text{s.t.}\quad\left(2\alpha_k\sum_{i\in I_k}\bar{c}_i\prod_{j=1}^{M}t_j^{a_{ij}}+\phi^{-1}(z_k^+)^2\sum_{i\in I_k}\sigma_i^2\prod_{j=1}^{M}t_j^{2a_{ij}}\right)\times$$

$$\prod_{i\in I_k}\prod_{p\in I_k}\left(\frac{\bar{c}_i\bar{c}_p\prod_{j=1}^{M}t_j^{(a_{ij}+a_{pj})}}{\delta_{ip}}\right)^{-\delta_{ip}}\left(\frac{\alpha_k^2}{\delta_0}\right)^{-\delta_0}\leq 1, k=1,...,K, \tag{4.41}$$

$$\sum_{i\in I_k}\bar{c}_i\prod_{j=1}^{M}t_j^{a_{ij}}+\phi^{-1}(z_k^-)\sqrt{\sum_{i\in I_k}\sigma_i^2\sum_{j=1}^{M}t_j^{2a_{ij}}}-\beta_k\leq 0, k=1,...,K,$$

$$z_k^++z_k^--1\geq y_k, 0\leq z_k^-; z_k^+\leq 1, k=1,...,K, \tag{4.42}$$

$$\prod_{k=1}^{K}y_k\geq 1-\epsilon, 0\leq y_k\leq 1, k=1,...,K. \tag{4.43}$$

We apply a logarithmic transformation of the problem by introducing $r_j=\log(t_j)$, $j=$

$1, ..., M$, $x_k = \log(y_k)$, $k = 1, ..., K$. We obtain the following biconvex equivalent problem

$$\min_{r,x,z^+,z^-} \sum_{i\in I_0} \bar{c}_i \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\}, \tag{4.44}$$

$$\text{s.t.} \quad \prod_{i\in I_k}\prod_{p\in I_k}\left(\frac{\bar{c}_i\bar{c}_p}{\delta_{ip}}\right)^{-\delta_{ip}}\left(\frac{\alpha_k^2}{\delta_0}\right)^{-\delta_0} \times$$

$$\left(2\alpha_k \sum_{i\in I_k} \bar{c}_i \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\} + \phi^{-1}(z_k^+)^2 \sum_{i\in I_k} \sigma_i^2 \exp\left\{\sum_{j=1}^{M} 2a_{ij} r_j\right\}\right)$$

$$\exp\left\{\sum_{i\in I_k}\sum_{p\in I_k} -\delta_{ip}\sum_{j=1}^{M}(a_{ij}+a_{pj})r_j\right\} \le 1, k = 1, ..., K, \tag{4.45}$$

$$\sum_{i\in I_k} \bar{c}_i \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\} +$$

$$\phi^{-1}(z_k^-)\sqrt{\sum_{i\in I_k} \sigma_i^2 \exp\left\{\sum_{j=1}^{M} 2a_{ij} r_j\right\}} - \beta_k \le 0, k = 1, ..., K, \tag{4.46}$$

$$\log(1-\epsilon) - \sum_{k=1}^{K} x_k \le 0, x_k \le 0, k = 1, ..., K, \tag{4.47}$$

$$\exp(x_k) - z_k^+ - z_k^- + 1 \le 0, k = 1, ..., K, \tag{4.48}$$

$$z_k^+ - 1 \le 0, k = 1, ..., K, \tag{4.49}$$

$$z_k^- - 1 \le 0, k = 1, ..., K, \tag{4.50}$$

$$-z_k^+ \le 0, k = 1, ..., K, \tag{4.51}$$

$$-z_k^- \le 0, k = 1, ..., K. \tag{4.52}$$

Let $z = (z^+, z^-)^T$, $f(r) = \sum_{i\in I_0} \bar{c}_i \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\}$, $h(x) = \left(\log(1-\epsilon) - \sum_{k=1}^{K} x_k, x_1, \ldots, x_K\right)^T$,
$l(x,z) = \left(\exp(x_1) - z_1^+ - z_1^- + 1, ..., \exp(x_K) - z_K^+ - z_K^- + 1\right)^T$,
$w(z) = (z_1{}^+ - 1, \ldots, z_K{}^+ - 1, z_1{}^- - 1, \ldots, z_K{}^- - 1, -z_1{}^+, \ldots, -z_K{}^+, -z_1{}^-, \ldots, -z_K{}^-)^T$
and

$$g(r,z)=\begin{pmatrix}
\prod_{i\in I_1}\prod_{p\in I_1}\left(\frac{\bar{c}_i\bar{c}_p}{\delta_{ip}}\right)^{-\delta_{ip}}\left(\frac{\alpha_1^2}{\delta_0}\right)^{-\delta_0}(2\alpha_1\sum_{i\in I_1}\bar{c}_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\}+\phi^{-1}(z_1^+)^2\times \\
\sum_{i\in I_1}\sigma_i^2\exp\left\{\sum_{j=1}^{M}2a_{ij}r_j\right\})\exp\left\{\sum_{i\in I_1}\sum_{p\in I_1}-\delta_{ip}\sum_{j=1}^{M}(a_{ij}+a_{pj})r_j\right\}-1 \\
\vdots \\
\prod_{i\in I_K}\prod_{p\in I_K}\left(\frac{\bar{c}_i\bar{c}_p}{\delta_{ip}}\right)^{-\delta_{ip}}\left(\frac{\alpha_K^2}{\delta_0}\right)^{-\delta_0}(2\alpha_K\sum_{i\in I_K}\bar{c}_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\}+\phi^{-1}(z_K^+)^2\times \\
\sum_{i\in I_K}\sigma_i^2\exp\left\{\sum_{j=1}^{M}2a_{ij}r_j\right\})\exp\left\{\sum_{i\in I_K}\sum_{p\in I_K}-\delta_{ip}\sum_{j=1}^{M}(a_{ij}+a_{pj})r_j\right\}-1 \\
\sum_{i\in I_1}\bar{c}_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\}+\phi^{-1}(z_1^-)\sqrt{\sum_{i\in I_1}\sigma_i^2\exp\left\{\sum_{j=1}^{M}2a_{ij}r_j\right\}}-\beta_1 \\
\vdots \\
\sum_{i\in I_K}\bar{c}_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\}+\phi^{-1}(z_K^-)\sqrt{\sum_{i\in I_K}\sigma_i^2\exp\left\{\sum_{j=1}^{M}2a_{ij}r_j\right\}}-\beta_K
\end{pmatrix},$$ we write (4.44)-(4.52) as

$$\begin{aligned}
&\min_{r,x,z} f(r),\\
&\text{s.t. } g(r,z)\le 0,\\
&\quad h(x)\le 0,\\
&\quad l(x,z)\le 0,\\
&\quad w(z)\le 0.
\end{aligned}\tag{4.53}$$

4.2.2 . Optimality conditions

Now, we study the optimality conditions for problem (4.53). Since the problem is bi-convex, we do not talk about a KKT system but rather a partial KKT system [69].

Definition 13. Let $(r^*,z^*,x^*)\in\mathbb{R}^M\times\mathbb{R}^{2K}\times\mathbb{R}^K$, if there exist $\mu^{(1)}$, $\mu^{(2)}$, $\lambda^{(1)}$, $\lambda^{(2)}$, γ and ω such that

$$\nabla f(r^*)+{\mu^{(1)}}^T\nabla_r g(r^*,z^*)=0, \tag{4.54}$$

$$\mu^{(1)}\ge 0,{\mu^{(1)}}^T g(r^*,z^*)=0, \tag{4.55}$$

$${\mu^{(2)}}^T\nabla_z g(r^*,z^*)+{\lambda^{(1)}}^T\nabla_z l(x^*,z^*)+\gamma^T\nabla_z w(z^*)=0, \tag{4.56}$$

$$\mu^{(2)}\ge 0,{\mu^{(2)}}^T g(r^*,z^*)=0,\lambda^{(1)}\ge 0,{\lambda^{(1)}}^T l(x^*,z^*),\gamma\ge 0,\gamma^T w(z^*)=0, \tag{4.57}$$

$${\lambda^{(2)}}^T\nabla_x l(x^*,z^*)+\omega^T\nabla_x h(x^*)=0, \tag{4.58}$$

$$\lambda^{(2)}\ge 0,{\lambda^{(2)}}^T l(x^*,z^*)=0,\omega\ge 0,\omega^T h(x^*)=0 \tag{4.59}$$

Then (r^*,z^*,x^*) is a partial KKT point of (4.53).

The following theorem is driven by the equivalence between a partial optimum and a partial KKT point of a biconvex program.

Theorem 26. Let $(r^*, z^*, x^*) \in \mathbb{R}^M \times \mathbb{R}^{2K} \times \mathbb{R}^K$ be a partial optimum of (4.53), with respect to partial Slater constraints qualification [69] at (r^*, x^*). Then (r^*, z^*, x^*) is a KKT point of (4.53) if and only if the partial KKT system (4.54)-(4.59) holds with $\mu^{(1)} = \mu^{(2)}$ and $\lambda^{(1)} = \lambda^{(2)}$. Furthermore, if $\mu^{(1)} = \mu^{(2)}$ and $\lambda^{(1)} = \lambda^{(2)}$ then (r^*, z^*, x^*) is a KKT point of (4.53).

4.2.3 . A dynamical neural network approach

Based on the partial KKT system (4.54)-(4.59) obtained in the previous section, we construct a dynamical neural network system that converges to a partial KKT point of (4.53). The dynamical neural network is driven by the following system, where r, z, x, μ, λ, γ, and ω are time-dependent variables

$$\frac{dr}{dt} = -(\nabla f(r) + \nabla_r g(r,z)^T(\mu + g(r,z))_+), \tag{4.60}$$
$$\frac{dz}{dt} = -(\nabla_z g(r,z)^T(\mu + g(r,z))_+ \nabla_z l(x,z)^T(\lambda + l(x,z))_+ + \nabla_z w(z)^T(\gamma + w(z))_+), \tag{4.61}$$
$$\frac{dx}{dt} = -(\nabla_x l(x,z)^T(\lambda + l(x,z))_+ \nabla_x h(x)^T(\omega + h(x))_+), \tag{4.62}$$
$$\frac{d\mu}{dt} = (\mu + g(r,z))_+ - \mu, \tag{4.63}$$
$$\frac{d\lambda}{dt} = (\lambda + l(x,z))_+ - \lambda, \tag{4.64}$$
$$\frac{d\gamma}{dt} = (\gamma + w(z))_+ - \gamma, \tag{4.65}$$
$$\frac{d\omega}{dt} = (\omega + h(x))_+ - \omega. \tag{4.66}$$

where $(x)_+ = \max(x, 0)$. For the sake of simplicity, let $y = (r, z, x, \mu, \lambda, \gamma, \omega)$ we rewrite the dynamical system (4.60)-(4.66) equivalently as follows

$$\begin{cases} \frac{dy}{dt} = \Phi(y) \\ y(t_0) = y_0 \end{cases}.$$

A generalized circuit implementation of the neural network (4.60)-(4.66) is provided in Figure 4.6.

To study the stability and the convergence of the proposed neural network, we first show the equivalence between a KKT point (4.54)-(4.59) and an equilibrium point of (4.60)-(4.66).

Lemma 27. Let x and y two n-dimensional vectors, we have

$$(x + y)_+ = x \Leftrightarrow \left(x \geq 0, y \leq 0 \text{ and } x^T y = 0\right).$$

Proof. $(x+y)_+ = x$ means that $x \geq 0$. Furthermore, $(x+y)_+ = x$ implies that $(x^Tx + x^Ty)_+ = x^Tx$ which means that $x^Ty = 0$ since $x^Tx \neq 0$. Remember that x and y are n-dimensional vectors, then $(x+y)_+ = x$ means that $\forall\, 1 \leq i \leq n\ (x_i + y_i)_+ = x_i$. If $x_i = 0$ then $(y_i)_+ = 0$ which means $y_i \leq 0$, else if $x_i > 0$ then $y_i = 0$. In conclusion $y \leq 0$. The conclusion follows. □

Theorem 28. Let $y = (r, z, x, \mu, \lambda, \gamma, \omega) \in \mathbb{R}^M \times \mathbb{R}^{2K} \times \mathbb{R}^K \times \mathbb{R}^{2K} \times \mathbb{R}^{K+1} \times \mathbb{R}^K \times \mathbb{R}^{4K}$, y is an equilibrium point of (4.60)-(4.66) if and only if (r, z, x) is a KKT point of (4.53).

Proof. Let $(r, z, x, \mu, \lambda, \gamma, \omega)$ an equilibrium point of (4.60)-(4.66), then $\frac{dr}{dt} = 0$, $\frac{dz}{dt} = 0$, $\frac{dx}{dt} = 0$, $\frac{d\mu}{dt} = 0$, $\frac{d\lambda}{dt} = 0$, $\frac{d\gamma}{dt} = 0$ and $\frac{d\omega}{dt} = 0$.
By Lemma 27, $\frac{d\mu}{dt} = 0 \Leftrightarrow (\mu + g(r,z))_+ = \mu \Leftrightarrow \mu \geq 0$ and $g(r,z) \leq 0$ and $\mu^T g(r,z) = 0 \Leftrightarrow$ (4.56) . We use the same approach to obtain (4.57) and (4.59).
$\frac{dr}{dt} = 0 \Leftrightarrow \nabla f(r) + \nabla_r g(r,z)^T(\mu + g(r,z))_+ = 0 \Leftrightarrow \nabla f(r) + \nabla_r g(r,z)^T\mu = 0 \Leftrightarrow$ (4.54) .
We obtain (4.56) and (4.58) following the same steps. We conclude that (r, z, x) is a partial KKT system of (4.53). It is easy to check the converse part of the theorem. □

Now, to show the stability and the convergence of our neural network, we need first to prove the negative semidefiniteness of the Jacobian matrix $\nabla\Phi(y)$.

Theorem 29. The Jacobian matrix $\nabla\Phi(y)$ is negative semidefinite.

Proof. Let $p, q, s, t \in \mathbb{N}$ such that
$(\mu + g)_+ = (\mu_1 + g_1(r,z), \mu_2 + g_2(r,z), ..., \mu_p + g_p(r,z), \underbrace{0, ..., 0}_{2K-p})$,
$(\lambda + l)_+ = (\lambda_1 + l_1(x,z), \lambda_2 + l_2(x,z), ..., \lambda_q + l_q(x,z), \underbrace{0, ..., 0}_{K-q})$,
$(\gamma + w)_+ = (\gamma_1 + w_1(z), \gamma_2 + w_2(z), ..., \gamma_s + w_s(z), \underbrace{0, ..., 0}_{4K-s})$,
$(\omega + h)_+ = (\omega_1 + h_1(x), \omega_2 + h_2(x), ..., \omega_t + h_t(x), \underbrace{0, ..., 0}_{K+1-t})$.

We write $\nabla\Phi(z) = \begin{bmatrix} A_1 & A_2 & A_3 & A_4 & A_5 & A_6 & A_7 \\ B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 \\ C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 \\ D_1 & D_2 & D_3 & D_4 & D_5 & D_6 & D_7 \\ E_1 & E_2 & E_3 & E_4 & E_5 & E_6 & E_7 \\ F_1 & F_2 & F_3 & F_4 & F_5 & F_6 & F_7 \\ G_1 & G_2 & G_3 & G_4 & G_5 & G_6 & G_7 \end{bmatrix}$,
where,

$$A_1 = -(\nabla^2 f(r) + \sum_{i=1}^{p}((\mu_i + g_i)\nabla_r^2 g_i^p(r,z)) + \nabla_r g^p(r,z)^T \nabla_r g^p(r,z)),$$

$$A_2 = -(\sum_{i=1}^{p}((\mu_i + g_i)\nabla_z \nabla_r g_i^p(r,z)) + \nabla_z g^p(r,z)^T \nabla_r g^p(r,z)),$$

$$A_4 = -\nabla_r g^p(r,z)^T, \qquad A_3 = 0, \qquad A_5 = 0, \qquad A_6 = 0, \qquad A_7 = 0,$$

$$B_1 = -(\sum_{i=1}^{p}((\mu_i + g_i)\nabla_r \nabla_z l_i^p(r,z)) + \nabla_z g^p(r,z)^T \nabla_r g^p(r,z)),$$

$$B_2 = -(\sum_{i=1}^{p}((\mu_i + g_i)\nabla_z^2 g_i^p(r,z)) + \nabla_r g^p(r,z)^T \nabla_z g^p(r,z) + \sum_{i=1}^{q}((\lambda_i + l_i)\nabla_z^2 l_i^q(x,z)) + \nabla_z l^q(x,z)^T \nabla_z l^q(x,z) + \sum_{i=1}^{s}((\gamma_i + w_i)\nabla_z^2 w_i^s(z)) + \nabla_z w^s(z)^T \nabla_z w^s(z)),$$

$$B_3 = -(\sum_{i=1}^{q}((\lambda_i + l_i)\nabla_x \nabla_z l_i^q(x,z)) + \nabla_z l^q(x,z)^T \nabla_x l^q(x,z)),$$

$$B_4 = -\nabla_z g^p(r,z)^T, \quad B_5 = -\nabla_z l^q(x,z)^T, \quad B_6 = -\nabla_z w^s(z)^T, \quad B_7 = 0, \quad C_1 = 0,$$

$$C_2 = -(\sum_{i=1}^{q}((\lambda_i + l_i)\nabla_z \nabla_x l_i^q(x,z)) + \nabla_x l^q(x,z)^T \nabla_z l^q(x,z)),$$

$$C_3 = -(\sum_{i=1}^{q}((\lambda_i + l_i)\nabla_x^2 l_i^q(x,z)) + \nabla_x l^q(x,z)^T \nabla_x l^q(x,z) + \sum_{i=1}^{t}((\omega_i + h_i)\nabla_x^2 h_i^t(x)) + \nabla_x h^t(x)^T \nabla_x h^t(x)),$$

$$C_4 = 0, \qquad C_6 = 0, \qquad C_5 = -\nabla_x l^q(x,z)^T, \quad C_7 = -\nabla_x h^t(x)^T,$$
$$D_1 = \nabla_r g^p(r,z)^T, \quad D_2 = \nabla_z g^p(r,z)^T, \quad D_3 = 0,$$

$D_4 = S_p = -\begin{bmatrix} O_{p\times p} & O_{p\times(2K-p)} \\ O_{(2K-p)\times p} & I_{(2K-p)\times(2K-p)} \end{bmatrix}$, $D_5 = 0$, $D_6 = 0$,$D_7 = 0$,

$E_1 = 0$, $E_2 = \nabla_z l^q(x,z)^T$, $E_3 = \nabla_x l^q(x,z)^T$, $E_4 = 0$, $E_5 = S_q = -\begin{bmatrix} O_{q\times q} & O_{q\times(K-q)} \\ O_{(K-q)\times q} & I_{(K-q)\times(K-q)}, \end{bmatrix}$,

$E_6 = 0$, $E_7 = 0$,

$F_1 = 0$, $F_2 = \nabla_z w^s(z)^T$, $F_3 = 0$, $F_4 = 0$, $F_5 = 0$, $F_6 = S_s = -\begin{bmatrix} O_{s\times s} & O_{s\times(4K-s)} \\ O_{(4K-s)\times s} & I_{(4K-s)\times(4K-s)}, \end{bmatrix}$,

$F_7 = 0$,

$G_1 = 0, G_2 = 0, G_3 = \nabla_x h^t(x)^T, G_4 = 0, G_5 = 0, G_6 = 0, G_7 = S_t = -\begin{bmatrix} O_{t\times t} & O_{t\times(K+1-t)} \\ O_{(K+1-t)\times t} & I_{(K+1-t)\times(K+1-t)}, \end{bmatrix}$.

We rewrite the Jacobian matrix $\nabla\Phi$ as follows, $\nabla\Phi(z) = \begin{bmatrix} A_1 & B_1^T & 0 & A_4 & 0 & 0 & 0 \\ B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & 0 \\ 0 & B_3^T & C_3 & 0 & C_5 & 0 & C_7 \\ -A_4 & -B_4 & 0 & S_p & 0 & 0 & 0 \\ 0 & -B_5 & -C_5 & 0 & S_q & 0 & 0 \\ 0 & -B_6 & 0 & 0 & 0 & S_s & 0 \\ 0 & 0 & -C_7 & 0 & 0 & 0 & S_t \end{bmatrix}$,

We can represent $\nabla\Phi$ as $\nabla\Phi(z) = \begin{bmatrix} A & B \\ -B^T & C \end{bmatrix}$, where $A = \begin{bmatrix} A_1 & B_1^T & 0 \\ B_1 & B_2 & B_3 \\ 0 & B_3^T & C_3 \end{bmatrix}$, $B = \begin{bmatrix} A_4 & 0 & 0 & 0 \\ B_4 & B_5 & B_6 & 0 \\ 0 & C_5 & 0 & C_7 \end{bmatrix}$, and $C = \begin{bmatrix} S_p & 0 & 0 & 0 \\ 0 & S_q & 0 & 0 \\ 0 & 0 & S_s & 0 \\ 0 & 0 & 0 & S_t \end{bmatrix}$,

Since w and h are convex and twice differentiable, there follows that $\nabla_z^2 w_i^s(z)$ and $\nabla_x^2 h_i^t(x)$ are positive semidefinite. Furthermore, g and l are biconvex and twice differentiable, then we have $\nabla_z^2 g_i^p(r,z)$, $\nabla_z^2 l_i^q(x,z)$ $\nabla_x^2 l_i^q(x,z)$ are positive semidefinite [56]. It is easy to see that $\nabla_r g^p(r,z)^T \nabla_z g^p(r,z)$, $\nabla_z l^q(x,z)^T \nabla_z l^q(x,z)$ and $\nabla_x l^q(x,z)^T \nabla_x l^q(x,z)$ are positive semidefinite. We conclude that B_2 and C_3 are negative semidefinite and hence $\begin{bmatrix} B_2 & B_3 \\ B_3^T & C_3 \end{bmatrix}$ is negative semidefinite [48]. Following the same steps, we show that A_1 is negative semidefinite, and we conclude that A is negative semidefinite. We can easily see that C is negative semidefinite. We conclude that $\nabla\Phi$ is negative semidefinite. □

Theorem 30. The neural network (4.60)-(4.66) is stable and converges to $y^* = (r^*, z^*, x^*, \mu^*, \lambda^*, \gamma^*, \omega^*)$ where (r^*, z^*, x^*) is a KKT point of (4.53).

Before giving the proof of Theorem 30, we need to introduce the relationship between the monotocity of mapping and the semidefiniteness of its Jacobian matrix.

Definition 14. [117] A mapping $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is said to be monotonic if

$$(x-y)^T(F(x)-F(y)) \geq 0, \qquad \forall x, y \in \mathbb{R}^n$$

Lemma 31. [117] A differentiable mapping $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is monotonic, if and only if the Jacobian matrix $\nabla F(x)$, $\forall x \in \mathbb{R}^n$, is positive semidefinite.

Proof. of Theorem 30.
Let $y^* = (r^*, z^*, x^*, \mu^*, \lambda^*, \gamma^*, \omega^*)$ an equilibrium point of (4.60)-(4.66) and consider the Lyapunov function defined by $V_1(y) = ||\Phi(y)||^2 + \frac{1}{2}||y - y^*||^2$. We have that $\frac{dV_1(y)}{dt} \leq 0$. In fact, $\frac{dV_1(y)}{dt} = (\frac{d\Phi}{dt})^T\Phi + \Phi^T\frac{d\Phi}{dt} + (y-y^*)^T\frac{dy}{dt}$. Or since $\frac{d\Phi}{dt} = \nabla\Phi(y)\Phi(y)$, then $\frac{dV_1(y)}{dt} = \Phi^T(\nabla\Phi(y)^T + \nabla\Phi(y))\Phi + (y-y^*)^T\Phi(y)$. We use Theorem 29 and Lemma 31 to conclude. There follows that the neural network (6.47)-(4.66) is stable in the sense of Lyapunov.
Notice that $V_1(y) \geq \frac{1}{2}\|y-y^*\|^2$, consequently there exists a convergent subsequence $(y(t_k)_{k\geq 0})$ such that $lim_{k\longrightarrow\infty} y(t_k) = \tilde{y}$ and $\frac{dV_1(\tilde{y})}{dt} = 0$.
Starting from a certain y_0, we have by LaSalle's invariance principle that the neural network converges to the largest invariant set contained in M which is defined by $M = \{y(t) \| \frac{dV_1(y)}{dt} = 0\}$.
Observe that $\frac{dy}{dt} = 0 \Leftrightarrow \frac{dV_1(y)}{dt} = 0$, we have then $\tilde{y}$ is an equilibrium point of (4.60)-(4.66).
Let show now that the neural network converges to $\tilde{y}$. For this, we consider the following Lyapunov function $V_2(y) = \|\Phi(y)\|^2 + \frac{1}{2}\|y-\tilde{y}\|^2$.
We have that V_2 is continuously differentiable, $V_2(\tilde{y}) = 0$ and $lim_{k\longrightarrow\infty} y(t_k) = \tilde{y}$, then

$lim_{t\longrightarrow\infty}V_2(y(t)) = V_2(\tilde{y}) = 0$.
Additionally, since $\frac{1}{2}\|y-\tilde{y}\|^2 \leq V_2(y)$ then $lim_{t\longrightarrow\infty}\|y-\tilde{y}\| = 0$ and $lim_{t\longrightarrow\infty}y(t) = \tilde{y}$. There follows that the neural network converges to an equilibrium point $\tilde{y} = (\tilde{r}, \tilde{z}, \tilde{x}, \tilde{\mu}, \tilde{\lambda}, \tilde{\gamma}, \tilde{\omega})$ where $(\tilde{r}, \tilde{z}, \tilde{x})$ is a KKT point of (4.53). □

4.2.4 . Numerical experiments

In order to test the performances of our proposed neural network, we first consider a transportation problem. Then, in Subsection 4.2.6, we study a generalized shape optimization problem to analyze the behavior of the neural network for different size instances and we compare it to a state-of-the-art alternate convex search. All the numerical experiments are done using Python. To compute the partial derivatives and the Jacobians, we use the package autograd. To generate the random instances, we use the package numpy.random. The ODEs of the recurrent dynamical neural networks are solved using the function solve_ivp of scipy.integrate library. We run our algorithms on Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz.

4.2.5 . Minimizing transport cost problem

In order to shift Vm^3 grains from a warehouse to a factory, we can use an open rectangular box of length x_1 meters, of width x_2 meters, and of height x_3 meters see Figure 4.7. We use the parameters $c1, c2, c3$ and $c4$ for the bottom costs, for each side costs and each end costs, and for each round trip of the box costs, respectively. We aim to find the minimum cost of transporting Vm^3 of grain.
We use a transporter to carry the box into the truck. The floor area of the box x_1x_2 must be less than $\beta_{floor}A_{floor}$ and larger than $\alpha_{floor}A_{floor}$ in order to avoid wasting in capacity, where A_{floor} is the floor area and α_{floor} and β_{floor} are the minimum and the maximum occupancy rates, respectively. The same thing applies for the wall area $2x_1x_3 + 2x_2x_3$ which must be less than $\beta_{wall}A_{wall}$ and larger then $\alpha_{wall}A_{wall}$. We assume that the floor and the wall areas of the transporter are random.

We reformulate then our minimization problem as

$$\begin{aligned} \min_{x\in\mathbb{R}^3_{++}} \quad & c_1x_1x_2 + 2c_2x_1x_3 + 2c_3x_2x_3 + c_4\frac{V}{x_1x_2x_3}, \\ \text{s.t.} \quad & \mathbb{P}(\alpha_{wall}A_{wall} \leq 2x_1x_3 + 2x_2x_3 \leq \beta_{wall}A_{wall}, \\ & \alpha_{floor}A_{floor} \leq x_1x_2 \leq \beta_{floor}A_{floor}) \geq 1-\epsilon. \end{aligned} \tag{4.67}$$

To solve problem (4.67) using our proposed neural network, we set $\alpha_{wall} = \alpha_{floor} = 50\%$, $\beta_{wall} = \beta_{floor} = 95\%$, $c_1 = 80$, $c_2 = 20$, $c_3 = 30$, $c_4 = 1$, $V = 80m^3$, $\frac{1}{A_{wall}} \sim \mathcal{N}(1.0/6.0, 0.01)$ and $\frac{1}{A_{floor}} \sim \mathcal{N}(3.0, 0.01)$.
The neural network converges to a minimum of 260.81 at $x_1 = 0.68m$, $x_2 = 0.46m$ and $x_3 = 2.01m$. We follow the convergence of x_1, x_2 and x_3 in Figure 4.8.

4.2.6 . Stochastic shape optimization problem

In order to evaluate the performances of the proposed dynamical network on different instances, we introduce the following shape optimization problem taken from [87].

We remind that $A_{wall\,j}$ and A_{floor} are random and defined as in the previous Subsection. The generalized problem is defined as follows.

$$\begin{aligned} &\min_{x\in\mathbb{R}^M_{++}} \prod_{i=1}^{m} x_i^{-1}, \\ &\text{s.t.} \quad \mathbb{P}(\alpha_{wall} \leq \sum_{j=1}^{m-1}(\frac{m-1}{A_{wall\,j}} x_1 \prod_{i=2, i\neq j}^{m} x_i) \leq \beta_{wall}, \\ &\qquad \alpha_{floor} \leq \frac{1}{A_{floor}} \prod_{j=2}^{m} x_j \leq \beta_{floor}) \geq 1-\epsilon. \end{aligned} \tag{4.68}$$

For the sake of comparison, we additionally solve the problem with individual constraints.

$$\begin{aligned} &\min_{x\in\mathbb{R}^M_{++}} \prod_{i=1}^{m} x_i^{-1}, \\ &\text{s.t.} \quad \mathbb{P}\left(\alpha_{wall} \leq \sum_{j=1}^{m-1}(\frac{m-1}{A_{wall\,j}} x_1 \prod_{i=2, i\neq j}^{m} x_i) \leq \beta_{wall}\right) \geq 1-\epsilon, \\ &\qquad \mathbb{P}\left(\alpha_{floor} \leq \frac{1}{A_{floor}} \prod_{j=2}^{m} x_j \leq \beta_{floor}\right) \geq 1-\epsilon. \end{aligned} \tag{4.69}$$

For the numerical experiments, we set $\epsilon = 0.05$, $\frac{1}{A_{floor}} \sim \mathcal{N}(1.0/20.0, 0.01)$, $\frac{1}{A_{wall_j}} \sim \mathcal{N}(1.0/60.0, 0.001)$, $\alpha_{wall} = 0.5$, $\beta_{wall} = 1.0$, $\alpha_{floor} = 0.5$ and $\beta_{floor} = 1.0$.
In order to check the robustness of our approach, we generate a set of 100 scenarios of the stochastic constraints, and we visualize the number of violated scenarios (VS) for each problem.
The numerical results are represented in Table 4.5. Column one gives the number of variables m. Columns two, three, and four give the objective value of problem (4.69), the number of VS, and the corresponding CPU time in seconds, respectively. Columns five, six, and seven give the objective value of problem (4.68), the number of VS, and the corresponding CPU time, respectively.
We observe that the objective values of the two problems are relatively close. Nevertheless, the problem (4.68) covers better the risk region. In fact, we remark that the number of violated scenarios for $m = 5$ for problem (4.68) is equal to 3, whereas the number of violated scenarios for problem (4.69) is equal to 6 as shown in Figures 4.9 and 4.10.

Finally, we test the impact of the confidence level $1-\epsilon$. We solve then problems (4.68) and (4.69) for different value of ϵ and we fix $m = 5$. The obtained results are recapitulated in Table 4.6. We observe that as the value of ϵ increases, the value of the objective function decreases though the number of VS increases. In fact, a higher value of ϵ means a larger risk area and a less restrictive minimization problem. To the best of our knowledge, the only work dealing with rectangular programs with joint probabilistic constraints is

Table 4.5: Results of the generalized transportation problem for different values of m

m	Individual constraints			Joint constraints		
	Obj value	VS	CPU Time	Obj value	VS	CPU Time
3	0.039	3	35.27	0.042	0	25.82
5	0.117	6	61.60	0.120	3	92.20
7	0.230	4	86.36	0.236	2	80.32
10	0.440	3	203.84	0.456	1	169.57
15	0.909	5	444.26	0.907	5	531.83
20	1.384	4	1111.72	1.402	2	874.75

Table 4.6: Results of the generalized transportation problem for different values of ϵ

ϵ	Individual constraints			Joint constraints		
	Obj value	VS	CPU Time	Obj value	VS	CPU Time
0.1	0.115	8	39.34	0.121	1	66.67
0.15	0.113	7	39.44	0.115	5	38.02
0.2	0.112	28	74.80	0.114	15	44.32
0.3	0.109	31	49.87	0.111	24	49.84
0.4	0.107	48	44.21	0.109	39	52.20

[87]. They propose new convex approximations based on the variable transformation and piecewise linear approximation methods to come up with lower and upper bounds for the optimal solutions. Since in the worst case, our neural network converges to a partial optimum of the minimization problem, we converge then at worst to an upper bound of the optimal solution. The advantage of our approach is that we don't use any convex approximation to approximate the optimal solution though the neurodynamic approach takes more time to solve the stochastic rectangular programs as the size of the problem increases. Therefore, we must note that our approach doesn't replace the existing convex approximations but gives promising results and opens the way for a new vision of the joint probabilistic problems.

In order to evaluate the quality of the upper bound obtained using the dynamical neural network, we compare the solutions with the values given by the alternate convex search method in [57]. We set $\epsilon = 0.05$ and solve problem (4.68) for different values of m. The obtained results are presented in Figure 4.11. The orange and bleu curves represent the optimal solution obtained by the dynamical neural network and the alternate convex search method, respectively. We notice that the results of our approach are comparable or even relatively better than the alternate convex search.

4.3 . A case study : Maximizing Signal to Interference Noise Ratio for Massive MIMO

The content of this Chapter was accepted in The 19^{th} International Conference on Mobile Web and Intelligent Information Systems.

Massive Multiple Input Multiple Output (MaMIMO) is a promising technology that is gaining attention in the field of communication systems and the Internet of Things (IoT). It involves the utilization of a large number of antennas that can simultaneously transmit and receive signals, resulting in increased capacity and improved performance. MaMIMO is considered as a potential solution for 5G networks and is expected to succeed the existing 4G LTE and LTE-A technologies. The key advantage of MaMIMO lies in its ability to handle interference among a large number of antennas. By leveraging spatial multiplexing and beamforming techniques, MaMIMO enables higher connectivity and improved network efficiency. It can adapt to high-density environments with a large number of devices, making it suitable for scenarios such as crowded stadiums, urban areas, and IoT deployments. Furthermore, MaMIMO offers reduced transmission latency, which is crucial for real-time applications like augmented reality and virtual reality. It also addresses energy efficiency concerns by optimizing the use of resources, thereby aligning with green communications guidelines. Additionally, MaMIMO provides better signal quality and enhanced security by exploiting multiple signal paths and employing advanced signal processing algorithms. We propose in this section a stochastic geometric formulation to solve a problem of maximizing signal to interference noise ratio for massive MIMO.

4.3.1 . Stochastic geometric formulation

We consider a single cell area, as depicted in Figure 4.12, where there exists a group of users denoted by $\mathcal{U} = 1, ..., K$. Each user in this set employs a single antenna to receive data from the base station. On the other hand, the base station is equipped with T antennas. Our objective is to maximize the worst user's Signal to Interference Noise Ratio (SINR) while respecting some power constraints imposed on each user.
The SINR value for user i can be mathematically expressed as follows [5]

$$\text{SINR}_i = \frac{p_i |g_i^H g_i|^2}{\sum\limits_{j \in \mathcal{U}, j \neq i} p_j |g_i^H g_j|^2 + |\sigma_i|^2} \tag{4.70}$$

We formulate our optimization problem as follows

$$\max_{p \in \mathbb{R}_{++}^K} \min_{i \in \mathcal{U}} \frac{p_i |g_i^H g_i|^2}{\sum\limits_{j \in \mathcal{U}, j \neq i} p_j |g_i^H g_j|^2 + |\sigma_i|^2}, \tag{4.71}$$

$$\text{s.t} \quad P_{min} \leq p_i \leq P_{max}, \forall i \in \mathcal{U}, \tag{4.72}$$

where p_i represents the power assigned to user i from the base station. The terms $g_i \in \mathbb{C}^{T \times 1}$, $g_i^H \in \mathbb{C}^{1 \times T}$, and σ_i^2 correspond to the beam domain channel vector associated with user i, its Hermitian transpose, and the Additive White Gaussian Noise (AWGN), respectively. We assume that the AWGN follows an independent complex Gaussian distribution with zero mean and unit variance ($\sigma_i \sim \mathcal{CN}(0, 1)$). The channel vectors g_i and g_i^H represent the quasi-static independent and identically distributed Rayleigh fading channels, where each entry is a complex number. The variables P_{min} and P_{max} define the lower and upper bounds, respectively, for the power assigned to each user, ensuring that the power remains within a certain range.
Let $a_{ij} = |g_i^H g_j|^2 |g_i^H g_i|^{-2}$ and $b_i = |\sigma_i|^2 |g_i^H g_i|^{-2}$ and by introducing an additional variable w we rewrite (4.71)-(4.72) as

$$\max_{p \in \mathbb{R}_{++}^K, w \in \mathbb{R}_{++}} \quad w, \tag{4.73}$$

$$\text{s.t} \quad \sum_{j \in \mathcal{U}, j \neq i} a_{ij} p_j p_i^{-1} w + b_i p_i^{-1} w \leq 1, \forall i \in \mathcal{U}, \tag{4.74}$$

$$P_{min} \leq p_i \leq P_{max}, \forall i \in \mathcal{U}. \tag{4.75}$$

An equivalent geometric minimization problem is given by

$$\min_{p \in \mathbb{R}_{++}^K, w \in \mathbb{R}_{++}} \quad w^{-1}, \tag{4.76}$$

$$\text{s.t} \quad \sum_{j \in \mathcal{U}, j \neq i} a_{ij} p_j p_i^{-1} w + b_i p_i^{-1} w \leq 1, \forall i \in \mathcal{U}, \tag{4.77}$$

$$P_{min} \leq p_i \leq P_{max}, \forall i \in \mathcal{U}. \tag{4.78}$$

We consider the case where the coefficients a_{ij} and b_i are not completely known and follow independent normal distributions, i.e., $a_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$ and $b_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. In this

case, we replace the deterministic constraint (4.77) with the following chance constraint, as proposed in [4].

$$\mathbb{P}\left\{\sum_{j\in\mathcal{U},j\neq i} a_{ij}p_j p_i^{-1}w + b_i p_i^{-1}w \leq 1, \forall i \in \mathcal{U}\right\} \geq 1-\epsilon. \tag{4.79}$$

with $1-\epsilon$, $0 < \epsilon \leq 0.5$, is a given confidence level. The joint chance constraint, as opposed to individual constraints, is used because it ensures that the constraint as a whole is satisfied to a certain confidence level. While individual chance constraints are easier to solve, they only guarantee that each constraint is satisfied to a certain confidence level independently, without considering the overall satisfaction of the constraints.
Using the independence between the coefficients and by introducing auxiliary variables $y_i \in^+$, $i \in \mathcal{U}$, we give the following deterministic equivalent for the joint constraint (4.79) [132]

$$\sum_{j\in\mathcal{U},j\neq i} \mu_{ij}p_j p_i^{-1}w + \mu_i p_i^{-1}w + \phi^{-1}(y_i)\left\{\sqrt{\sum_{j\in\mathcal{U},j\neq i}\sigma_{ij}^2 p_j^2 p_i^{-2}w^2 + \sigma_i^2 p_i^{-2}w^2}\right\} \leq 1, \forall i \in \mathcal{U}, \tag{4.80}$$

$$\prod_{i\in\mathcal{U}} y_i \geq 1-\epsilon, \tag{4.81}$$

$$0 \leq y_i \leq 1, \forall i \in \mathcal{U}, \tag{4.82}$$

We write then (6.76)-(6.78) equivalently as

$$\begin{aligned}
\min_{p\in^K_{++}, w\in_{++}} \quad & w^{-1}, \\
\text{s.t} \quad & \sum_{j\in\mathcal{U},j\neq i} \mu_{ij}p_j p_i^{-1}w + \mu_i p_i^{-1}w + \\
& \phi^{-1}(y_i)\left\{\sqrt{\sum_{j\in\mathcal{U},j\neq i}\sigma_{ij}^2 p_j^2 p_i^{-2}w^2 + \sigma_i^2 p_i^{-2}w^2}\right\} \leq 1, \forall i \in \mathcal{U}, \\
& 1-\epsilon-\prod_{i\in\mathcal{U}} y_i \leq 0, \\
& -y_i \leq 0, y_i - 1 \leq 0, \forall i \in \mathcal{U}, \\
& P_{min} - p_i \leq 0, p_i - P_{max} \leq 0, \forall i \in \mathcal{U}.
\end{aligned} \tag{GP}$$

The obtained equivalent deterministic problem (GP) is nonconvex, we apply then the logarithmic transformation $r_i = \log(p_i)$, $x_i = \log(y_i)$, $\forall i \in \mathcal{U}$ and $t = log(w)$ and obtain the

following problem

$$\min \exp(-t), \tag{4.83}$$

$$\text{s.t} \sum_{j\in\mathcal{U},j\neq i} \mu_{ij}\exp(r_j - r_i + t) + \mu_i\exp(t - r_i) \tag{4.84}$$

$$+\phi^{-1}(e^{x_i})\left\{\sqrt{\sum_{j\in\mathcal{U},j\neq i}\sigma_{ij}^2\exp(2r_j - 2r_i + 2t) + \sigma_i^2\exp(2t - 2r_i)}\right\} \leq 1, \forall i \in \mathcal{U},$$

$$\log(1-\epsilon) - \sum_{i\in\mathcal{U}} x_i \leq 0, x_i \leq 0, i \in \mathcal{U}, \tag{4.85}$$

$$\log(P_{min}) - r_i \leq 0, r_i - \log(P_{max}) \leq 0, \forall i \in \mathcal{U}. \tag{4.86}$$

Let $z = (r, t)^T$, for the sake of simplicity we write the optimization problem as

$$\min f(z), \tag{4.87}$$

$$\text{s.t } g_i(z, x) \leq 0, \forall i \in \mathcal{U}, \tag{4.88}$$

$$l(x) \leq 0, h_i(x) \leq 0, i \in \mathcal{U}, \tag{4.89}$$

$$v_i(z) \leq 0, w_i(z) \leq 0, \forall i \in \mathcal{U}. \tag{4.90}$$

where $f(z) = \exp(-t)$,$l(x) = \log(1-\epsilon) - \sum_{i\in\mathcal{U}} x_i$, $h_i(x) = x_i$, $v_i(z) = \log(P_{min}) - r_i$, $w_i(z) = r_i - \log(P_{max})$ and

$$g_i(z,x) = \sum_{j\in\mathcal{U},j\neq i} \mu_{ij}\exp(r_j - r_i + t) + \mu_i\exp(t - r_i) + \phi^{-1}(e^{x_i})\times$$
$$\left\{\sqrt{\sum_{j\in\mathcal{U},j\neq i}\sigma_{ij}^2\exp(2r_j - 2r_i + 2t) + \sigma_i^2\exp(2t - 2r_i)}\right\} - 1$$

The dynamical neural network is given in this case by

$$\frac{dz}{dt} = -(\nabla_z f(z) + \nabla_z g(z,x)^T(\alpha + g(z,x))_+ + \nabla_z v(z)^T(\gamma + v(z))_+ + \nabla_z w(z)^T(\lambda + w(z))_+), \tag{4.91}$$

$$\frac{dx}{dt} = -(\nabla_x l(x)^T(\beta + l(x))_+ + \nabla_x g(z,x)^T(\alpha + g(z,x))_+ + \nabla_x h(x)^T(\zeta + h(x))_+), \tag{4.92}$$

$$\frac{d\alpha}{dt} = (\alpha + g(z,x))_+ - \alpha, \tag{4.93}$$

$$\frac{d\gamma}{dt} = (\gamma + v(z))_+ - \gamma, \tag{4.94}$$

$$\frac{d\lambda}{dt} = (\lambda + w(z))_+ - \lambda, \tag{4.95}$$

$$\frac{d\beta}{dt} = (\beta + l(x))_+ - \beta, \tag{4.96}$$

$$\frac{d\zeta}{dt} = (\zeta + h(x))_+ - \zeta. \tag{4.97}$$

K	Individual constraints	VS	Joint constraints	VS
2	5.45	16	5.99	6
3	6.87	21	7.43	9
5	35.57	39	36.99	8
7	48.98	53	50.43	11
10	39.40	62	41.68	10
15	82.30	84	85.21	12
20	113.65	82	117.33	10

Table 4.7: Individual constraints vs. Joint constraints for different values of K

where $\alpha^{(1)} = (\alpha_1^{(1)}, ..., \alpha_K^{(1)})^T$, $\alpha^{(2)} = (\alpha_1^{(2)}, ..., \alpha_K^{(2)})^T$, $\gamma = (\gamma_1, ..., \gamma_K)^T$, $\lambda = (\lambda_1, ..., \lambda_K)^T$, $\zeta = (\zeta_1, ..., \zeta_K)^T$, $g = (g_1, ..., g_K)^T$, $v = (v_1, ..., v_K)^T$, $w = (w_1, ..., w_K)^T$ and $h = (h_1, ..., h_K)^T$ are time-continuous vectors.

4.3.2 . Numerical analysis

For the numerical experiments, we set $P_{min} = 0.1$ and $P_{max} = 0.5$. The confidence level is $\epsilon = 0.1$. We generate complex vectors $g_i \in \mathbb{C}^{T\times 1}$ and $g_i^H \in \mathbb{C}^{1\times T}$ for each $i \in \mathcal{U}$ from an independent complex Gaussian distribution with zero mean and unit variance. Each of these vectors is then multiplied by a factor randomly chosen from the set $3.0, 4.0, 5.0, 7.0$. We also generate the parameter σ_i for each $i \in \mathcal{U}$ from an independent complex Gaussian distribution with zero mean and unit variance. The variables a_{ij} and b_i are computed as explained in Section 4.3.1. We assume that $\mu_{ij} = a_{ij}$ and $\mu_i = b_i$, and we vary the values of σ_{ij} and σ_i within the set $0.1, 0.2, 0.3$.

We first solve (GP) for $K = 5$ with different feasible initial points y_0 and observe the convergence process of the neural network for each case. As shown in Figure 4.13, we observe that the neural network converges to the same final value regardless of the starting point.

In order to demonstrate the advantage of using joint constraints instead of individual constraints to handle the uncertainty in constraints (4.77), we solve (GP) for different numbers of users, ranging from $K = 2$ to $K = 20$, using both joint and individual chance constraints. We generate 100 instances of the stochastic variables a_{ij} and b_i and track the number of times the constraints (4.77) are violated. The results are summarized in Table 4.7.

Column one represents the number of users K, while columns two and three present the optimal solution and the number of VS obtained using individual constraints, respectively. Similarly, columns four and five display the optimal solution and the number of VS obtained using joint constraints.

We observe that the number of VS when using individual constraints is consistently bigger than the number of VS when using joint constraints. This difference becomes more significant as the value of K increases. By employing joint chance constraints, we ensure a better coverage of the risk area and achieve a more robust solution.

K	Sequential Algorithm	VS	Neural Network	VS	GAP
2	5.40	12	5.10	6	5.88
3	25.77	10	25.61	8	0.62
5	28.97	11	28.88	9	0.31
7	68.79	10	68.56	8	0.33
10	70.81	21	69.68	14	1.62
15	84.43	7	84.39	6	0.04
20	117.37	13	117.33	10	0.03

Table 4.8: Neural network vs. the sequential algorithm for different values of K

For comparison purposes, we solve problem (GP) using both the neurodynamic approach and the sequential algorithm. The results obtained are summarized in Table 6.1. Column one represents the number of users K, while columns two and three provide the optimal solution and the number of VS obtained using the sequential algorithm, respectively. Similarly, columns four and five present the optimal solution and the number of VS obtained using the neurodynamic approach. Finally, column six displays the gap between the two solutions.
We observe that the neurodynamic approach yields better solutions compared to the sequential algorithm. Additionally, the number of violated scenarios for the solutions obtained using the neurodynamic approach is slightly lower than those obtained using the sequential algorithm.

We consider the case where $K = 5$ and vary the value of ϵ in the range of $[0.05, 0.4]$. The results obtained are summarized in Table 4.9.
We observe that as ϵ increases, the problem becomes less conservative. This means that the chance constraint becomes less strict, allowing for a higher probability of violating the constraint. Additionally, we observe that the gap between the two approaches (neurodynamic and sequential) increases as ϵ increases, as shown in Figure 4.14. This indicates that the neurodynamic approach provides better solutions compared to the sequential algorithm when dealing with larger values of ϵ.
Furthermore, we notice that the number of violated scenarios increases as ϵ increases, as shown in Figure 4.15. The difference in the number of violated scenarios becomes more significant as ϵ increases, indicating that the neurodynamic approach ensures better robustness in handling uncertainties.

ϵ	Sequential Algorithm	VS	Neural Network	VS	GAP
0.05	30.23	4	29.89	2	1.13
0.1	29.47	15	29.07	9	1.37
0.15	28.96	22	28.53	11	1.50
0.2	28.56	32	28.10	19	1.63
0.3	27.87	54	27.40	26	1.71
0.4	27.30	63	26.81	34	1.82

Table 4.9: Neural network vs. the sequential algorithm for different values of ϵ

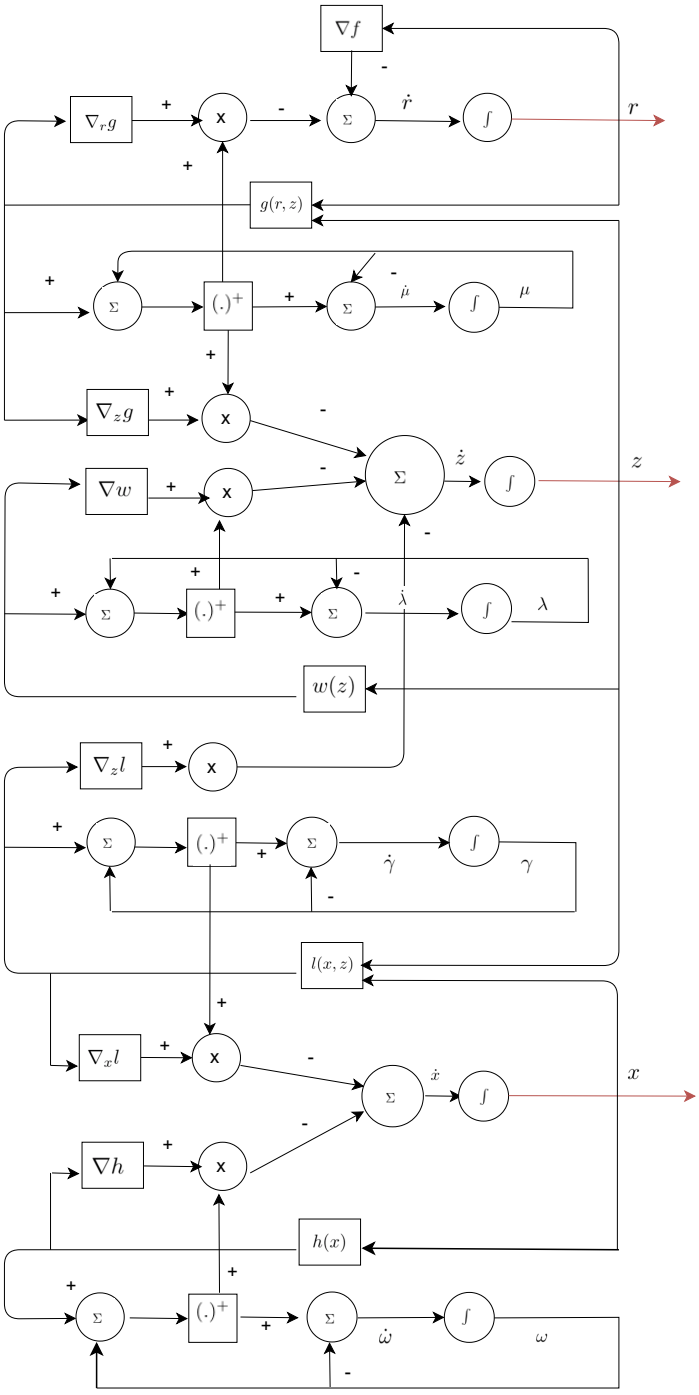

Figure 4.6: The architecture of the neural network (4.60)-(4.66)

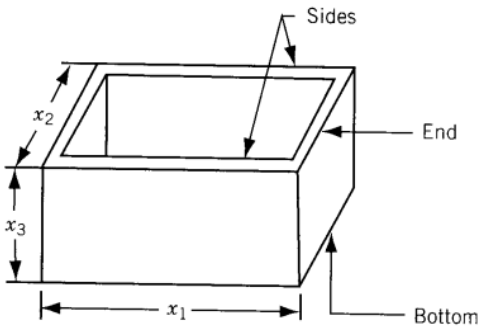

Figure 4.7: The shape of the box [1]

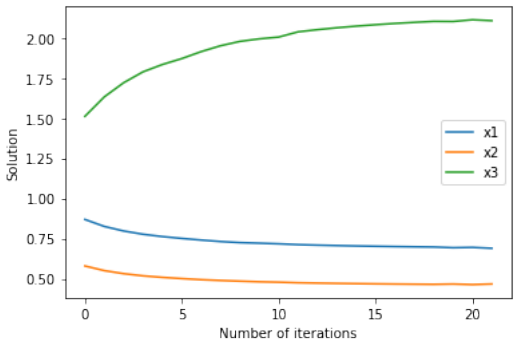

Figure 4.8: The convergence of the neural network of problem (4.67)

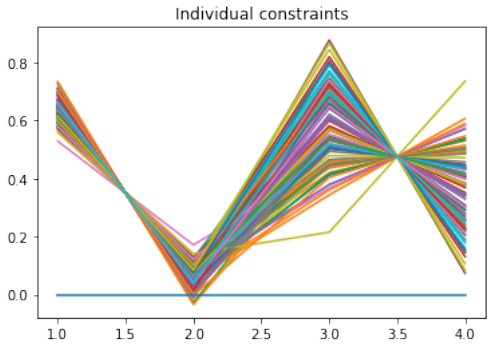

Figure 4.9: Out of 100 scenarios, the constraints were violated 6 times

.

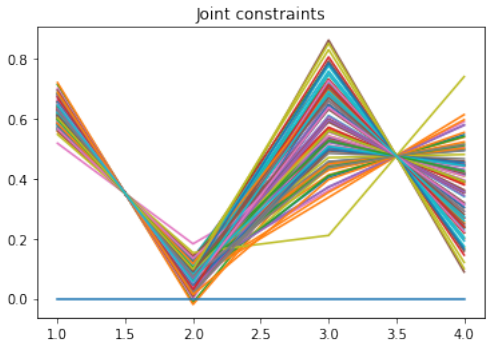

Figure 4.10: Out of 100 scenarios, the constraints were violated 3 times

.

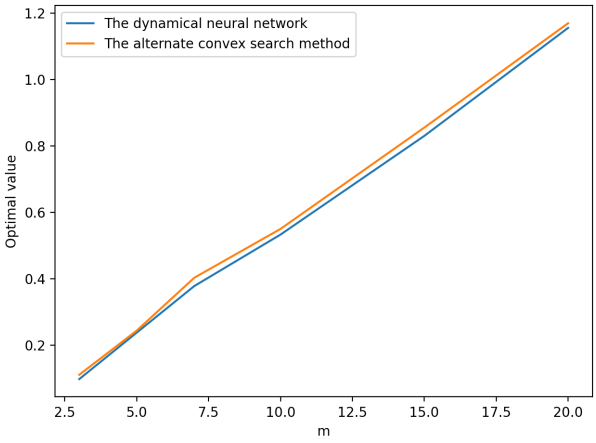

Figure 4.11: The Dynamical Neural Network vs. The Alternate Convex Search

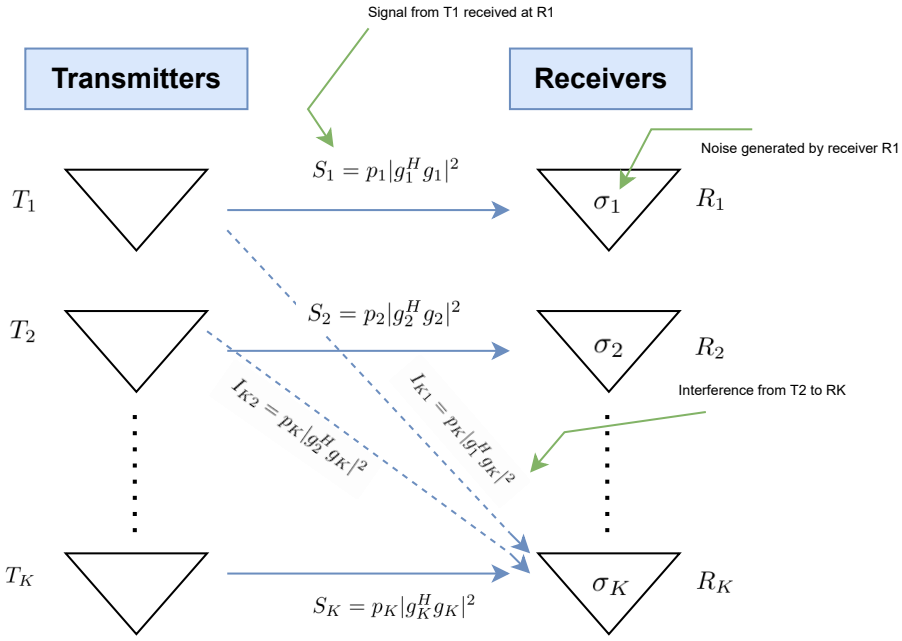

Figure 4.12: Signal to Interference plus Noise Ratio, illustration.

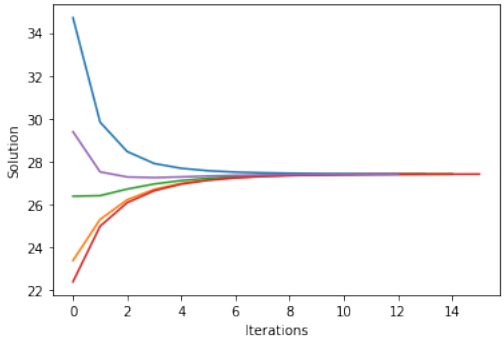

Figure 4.13: Convergence of the neural network different starting points y_0

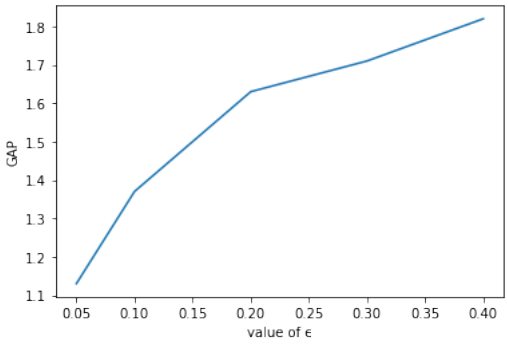

Figure 4.14: Evolution of GAP function to ϵ

.

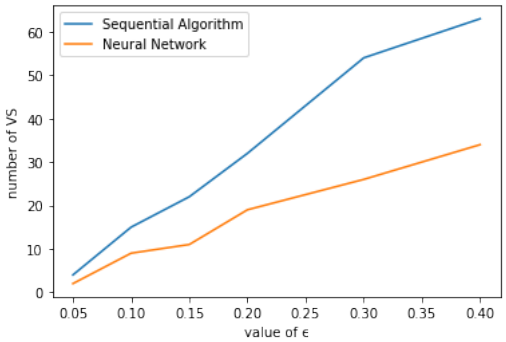

Figure 4.15: Evolution of VS function to ϵ

.

5 - Dependent Joint Chance Constrained Geometric Optimization

In this chapter, we delve into the application of copula theory for modeling dependencies in joint probabilistic constrained geometric programs with dependent rows. We also study in the second Section linear programs as a particular case. We focus on scenarios where the random variables are assumed to follow an elliptical distribution. To lay the foundation, we introduce the fundamental concepts of copula theory and refresh our understanding of the key properties associated with elliptical distributions. This groundwork will provide the necessary background for our subsequent analysis.

5.1 . Copula Theory

The use of copula theory in statistical modeling has gained significant interest in recent years. Copulas provide a means to capture the dependencies between random variables and derive joint distributions. The concept of copulas was introduced by Fisher in 1959 [126], and since then, it has found applications in various fields.
One of the key advantages of copulas is that they provide scale-free measures of statistical dependence. This means that the measure of dependence is independent of the marginal distributions of the variables involved. This property makes copulas useful in capturing and analyzing the dependence structure between variables.
The application of copulas extends to fields such as finance and economics. Jouini and Clemen [70] discuss the use of copulas in aggregating information from different sources by considering multivariate distributions that are functions of their marginals. Patton [105] provides a comprehensive review of the applications of copulas in finance and economics, highlighting their usefulness in modeling dependencies and risks.
In the context of joint chance-constrained optimization, Houda and Lisser [64] use copulas to model the dependence between random variables. Copulas offer a flexible framework to capture the joint distribution of uncertain variables and enable the formulation of joint chance constraints. Furthermore, Liu et al. [91] propose a collective neurodynamic approach using multiple interconnected recurrent neural networks for distributed constrained optimization. This approach leverages the power of neural networks to tackle optimization problems with complex constraints.

In this section, we introduce the notion of a copula and the correspondent properties we use in this paper.

Definition 15. A copula C : $[0,1]^d \longrightarrow [0,1]$ of dimension d is a joint cumulative distribution function with for which the marginals are uniformly distributed on the interval $[0,1]$.

Theorem 32. (Sklar's Theorem 1959) Let F a d-dimensional cumulative distribution func-

tion with marginals $\mathsf{F}_1, \mathsf{F}_2, .., \mathsf{F}_d$, there exists a copula C such that

$$\forall x \in^d \quad \mathsf{F}(x) = \mathsf{C}(\mathsf{F}_1(x_1), \mathsf{F}_2(x_2), .., \mathsf{F}_d(x_d)).$$

If $\mathsf{F}_1, \mathsf{F}_2, .., \mathsf{F}_d$ are all continuous then C is unique and given by

$$\mathsf{C}(u) = \mathsf{F}({\mathsf{F}_1}^{-1}(u_1), {\mathsf{F}_2}^{-1}(u_2), .., {\mathsf{F}_d}^{-1}(u_d)),$$

otherwise C is uniquely determined on $\text{rang}(\mathsf{F}_1) \times \text{rang}(\mathsf{F}_2), .., \text{rang}(\mathsf{F}_d)$.

Definition 16. If a copula C has a density, then it is expressed as follows

$$\mathsf{C}(u_1, u_2, .., u_d) = \frac{\partial^d \mathsf{C}(u_1, u_2, .., u_d)}{\partial u_1 u_2 .. u_d}$$

To capture the dependencies between stochastic parameters, we often rely on copulas. While many copulas lack explicit analytical expressions, such as the Gaussian copula, there exists a class of copulas known as Archimedean copulas that offer explicit mathematical formulations. Archimedean copulas derive their name from the Greek mathematician Archimedes and are defined based on specific generator functions. These generator functions characterize the underlying dependence structure of the copula.

Definition 17. A copula C is Archimedean if it is represented as follows

$$\mathsf{C}(u_1, u_2, .., u_d; \theta) = \psi^{[-1]}(\psi(u_1; \theta), \psi(u_2; \theta), .., \psi(u_d; \theta); \theta),$$

where $\psi : [0, 1] \times \Theta \longrightarrow [0, \infty)$ a strictly continuously declining function called generator of C, such that $\psi(1; \theta) = 0$. $\psi^{[-1]}$ is a pseudo-inverse of ψ defined by $\psi^{[-1]}(t; \theta) = \begin{cases} \psi^{-1}(t; \theta) \text{ if } 0 \leq t \leq \psi(0; \theta) \\ 0 \text{ if } t \geq \psi(0; \theta) \end{cases}$ and θ is a dependency parameter.

Definition 18. If $\lim_{u \to +\infty} \psi(u) = +\infty$, then C is called a strict Archimedean copula and ψ is called a strict generator.

Theorem 33. Let $\psi : [0, 1] \longrightarrow \mathbb{R}^+$ a convex, strictly decreasing function such that $\lim_{u \to +\infty} \psi(u) = +\infty$ and

$$(-1)^k \frac{d^k}{dt^k} \psi^{-1}(t) \geq 0, k = 0, 1, ..K. \tag{5.1}$$

Then ψ is a strict copula generator.

Theorem 34. All copula generators are convex.

We give in Table 5.1 some examples of Archimedean copulas often used in the literature.

Name of copula	Parameter θ	Generator $\psi_\theta(t)$
Frank	$\theta > 0$	$-\ln(\frac{e^{-\theta t}-1}{e^{-\theta}-1})$
Gumbel–Hougaard	$\theta \geq 1$	$e^{-t^{\frac{1}{\theta}}}$
Joe	$\theta > 1$	$-\ln(1-(1-t)^\theta)$

Table 5.1: Examples of some commonly used Archimedean copulas

5.2 . Elliptically symmetric random vectors

The concept of elliptically symmetric random vectors emerged in the field of probability theory in the early 1970s as an extension of the class of multivariate normal distributions. Extensive research on elliptical distributions and their fundamental properties has been conducted, and a comprehensive survey of these results can be found in the book by Fang et al. [44]. This survey provides a comprehensive overview of the basic concepts, key findings, and properties of elliptical distributions, making it an invaluable resource for further exploration and study in this area. We consider in this chapter that the random vectors are elliptically distributed. We then recall some definitions and properties of the elliptical family.

Definition 19. An n-dimensional vector X follows an elliptical symmetric distribution if there exists a function Ψ such that its characteristic function is given by

$$\phi(z) = \mathbb{E}[e^{iz^T X}] = e^{iz^T \mu}\Psi(z^T \Sigma z). \tag{5.2}$$

where μ is the location parameter, and Σ is a positive-definite matrix. The function Ψ is called a characteristic generator of the elliptical distribution. We note $X \sim \text{Ellip}(\mu, \Sigma, \phi)$.

Definition 20. When the density function of an elliptical distribution exists, it must have the structure

$$f(x) = \frac{C}{\sqrt{\det\Sigma}} g\left(\sqrt{(x-\mu)^T \Sigma^{-1}(x-\mu)}\right). \tag{5.3}$$

where $g : \mathbb{R}^+ \to \mathbb{R}^{++}$ is a so-called radial density and $c > 0$ is a normalization factor ensuring that f integrates to one.

Remark 4. Table 5.2 gives a selection of some multivariate elliptical distributions, together with their characteristic generators, radial densities, quantile functions of the standard distribution, and the convexity conditions of the quantile function. The quantile functions of the selected distributions are given in Figure 5.2.

5.3 . Geometric programs

5.3.1 . Deterministic equivalent formulation

Law	Characteristic generator	Radial density	Quantile Function	Convexity Condition
Normal	$e^{-\frac{1}{2}t}$	$e^{-\frac{1}{2}t^2}$	$\sqrt{2}\text{erf}^{-1}(2\alpha-1)$	$\frac{1}{2}\leq\alpha\leq 1$
Laplace	$(1+\frac{1}{2}t)^{-1}$	$e^{-\sqrt{2}\lvert t\rvert}$	$\ln(2\alpha)$, if $0\leq\alpha\leq\frac{1}{2}$ $-\ln(2(1-\alpha))$, if $\frac{1}{2}\leq\alpha\leq 1$	$\frac{1}{2}\leq\alpha\leq 1$
Cauchy	$e^{-\sqrt{t}}$	$(1+t^2)^{-\frac{n+1}{2}}$	$\tan(\pi(\alpha-\frac{1}{2}))$	$\frac{1}{2}\leq\alpha\leq 1$
Logistic	$\frac{2\pi\sqrt{t}}{e^{\pi\sqrt{t}}-e^{-\pi\sqrt{t}}}$	$\frac{e^{-t^2}}{(1+e^{-t^2})^2}$	$\ln(\frac{\alpha}{1-\alpha})$	$\frac{1}{2}\leq\alpha\leq 1$

Table 5.2: Table of selected elliptical distributions

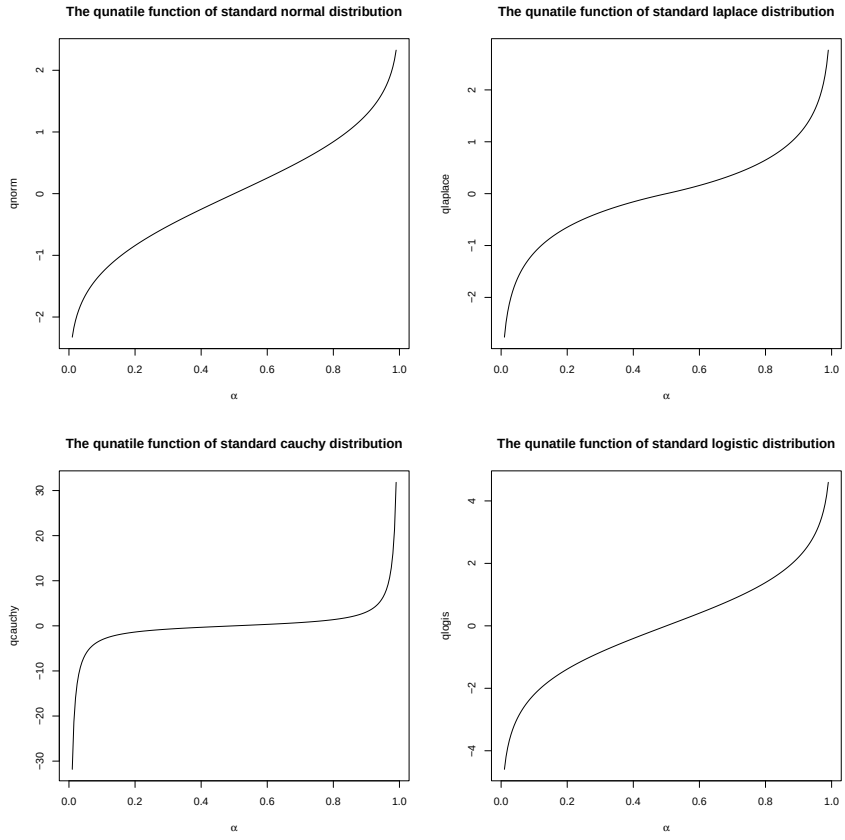

Figure 5.1: Quantile functions of selected standard elliptical distributions

In this section, we study the following joint chance-constrained geometric program

$$\begin{aligned} \min_{t\in\mathbb{R}^M_{++}} \ & \mathbb{E}\left[\sum_{i=1}^{I_0} c_i \prod_{j=1}^{M} t_j^{a_{ij}}\right], \\ \text{s.t} \ & \mathbb{P}\left(\sum_{i=1}^{I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \leq 1, k = 1, ..., K\right) \geq \alpha, \end{aligned} \tag{5.4}$$

with $\alpha \in [0.5, 1)$.

Assumption 35. We assume that the row vectors are dependent and elliptically distributed and that the dependence is captured by a Gumbel-Hougaard copula, i.e., $C_0 = (c_1, ..., c_{I_0}) \sim$ Ellip(μ_0, Σ_0, ϕ) and $C_k = (c_1, ..., c_{I_k}) \sim$ Ellip(μ_k, Σ_k, ϕ). Where μ_k is a mean vector and $\Sigma_k = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1I_k} \\ \vdots & \vdots & \dots & \vdots \\ \sigma_{I_k1} & \sigma_{I_k2} & \dots & \sigma_{I_kI_k} \end{bmatrix}$ is a positive definite matrix, $k = 1, ..., K$.

Assumption 36. In this research, we consider only the elliptical distributions present in Table 5.2.

Using the theory of copula and Sklar's properties, a deterministic equivalent of (5.4) is given by [27]

$$\begin{aligned} \min_{t\in\mathbb{R}^M_{++}} \ & \sum_{i=1}^{I_0} \mu_i \prod_{j=1}^{M} t_j^{a_{ij}}, \\ \text{s.t} \ & \sum_{i=1}^{I_k} \mu_i \prod_{j=1}^{M} t_j^{a_{ij}} + \phi^{-1}((\alpha^{y_k})^{\frac{1}{\theta}}) \sqrt{\sum_{i=1}^{I_k}\sum_{l=1}^{I_k} \sigma_{il} \prod_{j=1}^{M} t_j^{a_{ij}+a_{lj}}} \leq 1, \ k = 1, ..., K, \\ & \prod_{k=1}^{K} y_k \geq \alpha, \ 0 < y_k \leq 1 \text{ , } k = 1, ..., K, \end{aligned} \tag{5.5}$$

where ϕ is one-dimensional standard elliptical distribution.

Problem (5.5) is not convex, so we apply a logarithmic transformation by setting $t_j =$

e^{r_j}. This transformation allows us to reformulate problem (5.5) as follows

$$
\begin{aligned}
&\min \sum_{i=1}^{I_0} \mu_i \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\}, \\
&\text{s.t } \sum_{i=1}^{I_k} \mu_i \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\} + \phi^{-1}((\alpha^{y_k})^{\frac{1}{\theta}}) \sqrt{\sum_{i\in I_k}\sum_{l\in I_k} \sigma_{il} \exp\left\{\sum_{j=1}^{M}(a_{ij}+a_{lj}) r_j\right\}} \leq 1,\ k=1,...,K, \\
&\quad \sum_{k=1}^{K} -\ln(y_k) \leq -\ln(\alpha), \\
&\quad -y_k < 0\ , k=1,...,K, \\
&\quad y_k \leq 1\ , k=1,...,K.
\end{aligned}
\tag{5.6}
$$

Theorem 37. The resulting problem (5.6) is biconvex.

Proof. The convexity on r is straightforward. We have for Gumbel-Hougaard copula that $\theta \geq 1$, there follows that $0 < \frac{1}{\theta} \leq 1$. Since $0 < y_k \leq 1$, we obtain $0 < {y_k}^{\frac{1}{\theta}} \leq 1$. We have then, $(\alpha^{y_k})^{\frac{1}{\theta}} \geq \alpha \geq 0.5$. From Table 5.2, $y_k \mapsto \phi^{-1}((\alpha^{y_k})^{\frac{1}{\theta}})$ is convex. The conclusion follows. □

Theorem 38. Let $(r^*, y^*) \in \mathbb{R}^M \times \mathbb{R}^K$ and $g_k(r, y_k) = \phi^{-1}((\alpha^{y_k})^{\frac{1}{\theta}}) \sqrt{\sum_{i\in I_k}\sum_{l\in I_k} \sigma_{kil} \exp\left\{\sum_{j=1}^{M}(a_{kji}+a_{kjt}) r_j\right\}}$. If there exist γ_r, γ_y, β_+, β_- and λ such that

$$
\begin{bmatrix} \sum_{i\in I_0} \mu_i a_{i1} \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\} \\ \vdots \\ \sum_{i\in I_0} \mu_i a_{iM} \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\} \end{bmatrix} + \sum_{k=1}^{K} \gamma_{rk} \begin{bmatrix} \sum_{i\in I_k} a_{i1} \exp\left\{\sum_{j=1}^{M} \mu_i a_{ij} r_j\right\} + \nabla_{r_1} g_k(r, y_k) \\ \vdots \\ \sum_{i\in I_k} \mu_i a_{iM} \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\} + \nabla_{r_M} g_k(r, y) \end{bmatrix} = 0,
$$

$$
\sum_{i\in I_k} \gamma_{y_k} \nabla_{y_k} g_k(r, y) + \beta_{+k} - \beta_{-k} - \frac{\lambda}{y_k} = 0, k = 1, ..., K \tag{5.7}
$$

$$
\gamma_r \geq 0,\ \sum_{i\in I_k} \gamma_{rk} \left\{\sum_{i\in I_k} \mu_i \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\} + g_k(r, y)\right\} = 0,\ \gamma_y \geq 0,\ \sum_{i\in I_k} \gamma_{y_k} \left\{\sum_{i\in I_k} \mu_i \exp\left\{\sum_{j=1}^{M} a_{ij} r_j\right\} + g_k(r, y)\right.
$$

$$
\beta_+ \geq 0,\ \beta_{+k}(y_k - 1) = 0,\ \beta_- \geq 0,\ \beta_{-k}(y_k) = 0,\ \lambda \geq 0,\ \lambda \geq 0,\ \lambda(\ln(\alpha) - \sum_{k=1}^{K} \ln(y_k)) = 0,\ k = 1, ..., K
$$

then (r^*, y^*) is a partial KKT point of (5.5).

Let $(r^*, y^*) \in \mathbb{R}^M \times \mathbb{R}^K$. We recall that if (5.5) is satisfied with partial Slater constraint qualification at (r^*, y^*), then (r^*, y^*) is a partial optimum of (5.5) if and only if (r^*, y^*) is a partial KKT point of (5.5). Furthermore if $\gamma_r = \gamma_y$, then (r^*, y^*) is a KKT point of (5.5).

5.3.2 . The dynamical neural network

We propose the following dynamical neural network to solve program (5.5)

$$\frac{dr_j}{dt} = -\left(\sum_{i\in I_0} a_{ij}\exp\left\{\sum_{j=1}^{M} a_{ij}r_j\right\} + \sum_{k=1}^{K}\left(\gamma_k + \left\{\sum_{i\in I_k}\mu_i\exp\left\{\sum_{j=1}^{M} a_{ij}r_j\right\} + g_k(r,y)\right\}\right)_+ \right.$$
$$\left. \times \left(\sum_{i\in I_k} a_{ij}\mu_i\exp\left\{\sum_{j=1}^{M} a_{ij}r_j\right\} + \nabla_{r_j} g_k(r,y_k)\right)\right), j = 1,...,M,$$

$$\frac{dy_k}{dt} = -\left(\sum_{i\in I_k}\nabla_{y_k} g_k(r,y) \times \left(\gamma_k + \left\{\sum_{i\in I_k}\mu_i\exp\left\{\sum_{j=1}^{M} a_{ij}r_j\right\} + g_k(r,y)\right\}\right)_+ \right.$$
$$\left. + \beta_{+k}(\beta_{+k} + y_k - 1)_+ + \beta_{-k}(\beta_{-k} - y_k)_+ - \frac{\lambda}{y_k}\Big(\lambda + \ln(\alpha) - \sum_{k=1}^{K}\ln(y_k)\Big)_+\right), k = 1,...,K,$$

$$\frac{d\gamma_k}{dt} = \left(\gamma_k + \left\{\sum_{i\in I_k}\mu_i\exp\left\{\sum_{j=1}^{M} a_{ij}r_j\right\} + g_k(r,y)\right\}\right)_+ - \gamma_k, k = 1,...,K, \tag{DNN}$$

$$\frac{d\beta_{+k}}{dt} = \Big(\beta_{+k} + y_k - 1\Big)_+ - \beta_{+k}, k = 1,...,K,$$

$$\frac{d\beta_{-k}}{dt} = \Big(\beta_{-k} - y_k\Big)_+ - \beta_{-k}, k = 1,...,K,$$

$$\frac{d\lambda}{dt} = \Big(\lambda + \ln(\alpha) - \sum_{k=1}^{K}\ln(y_k)\Big)_+ - \lambda.$$

Let $r = (r_1,...,r_M)^T$, $y = (y_1,...,r_K)^T$, $\gamma = (\gamma_1,...,\gamma_K)^T$, $\beta_+ = (\beta_{+1},...,\beta_{+K})^T$ and $\beta_- = (\beta_{-1},...,\beta_{-K})^T$. For the sake of simplicity, we can write (DNN) as

$$\frac{dr}{dt} = A(r,y,\gamma), \tag{5.8}$$

$$\frac{dy}{dt} = B(r,y,\gamma,\beta_+,\beta_-,\lambda), \tag{5.9}$$

$$\frac{d\gamma}{dt} = C(r,y,\gamma), \tag{5.10}$$

$$\frac{d\beta_+}{dt} = D(y,\beta_+), \tag{5.11}$$

$$\frac{d\beta_-}{dt} = E(y,\beta_-), \tag{5.12}$$

$$\frac{d\lambda}{dt} = F(y,\lambda). \tag{5.13}$$

A simplified circuit implementation of (5.8)-(5.13) is given in Figure 5.2.

Theorem 39. Suppose that (r^*,y^*) is a partial optimum of (5.6) and $\gamma^*,\beta_+^*,\beta_-^*$ and λ^* the corresponding Lagrange multipliers, then $(r^*,y^*,\gamma^*,\beta_+^*,\beta_-^*,\lambda^*)$ is a an equilibrium point of (DNN). Additionally, every equilibrium point of (DNN) is a KKT point of (5.6).

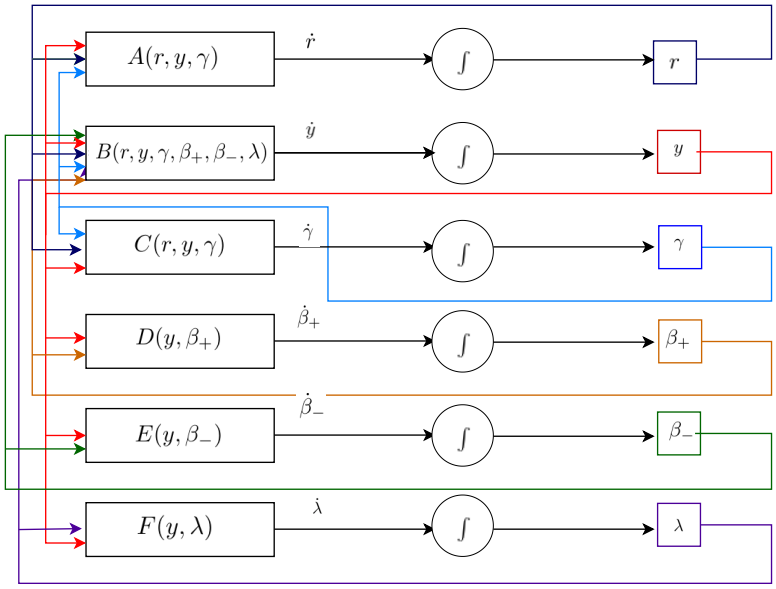

Figure 5.2: A simplified implementation of the neural network (5.8)-(5.13)

Proof. Let $(r^*, y^*, \gamma^*, \beta_+^*, \beta_-^*, \lambda^*)$ is a an equilibrium point of (DNN), then all the left side derivatives in (DNN) are equal to zero. We use the fact that $((a+b)_+ = a) \Leftrightarrow \big(a \geq 0, b \leq 0 \text{ and } a^T b = 0\big)$ and we obtain the partial KKT system (5.7) with $\gamma_r = \gamma_y = \gamma$. Now if (r^*, y^*) is a partial optimum of (5.5) and $\gamma^*, \beta_+^*, \beta_-^*$ and λ^* the corresponding Lagrange multipliers, then it is straightforward that $(r^*, y^*, \gamma^*, \beta_+^*, \beta_-^*, \lambda^*)$ is a an equilibrium point of (DNN). □

Theorem 40. The neurodynamic model (DNN) is stable and globally convergent to a KKT point $(r^*, y^*, \gamma^*, \beta_+^*, \beta_-^*, \lambda^*)$ of (5.6).

Proof. Let $\mu = (r, y, \gamma, \beta_+, \beta_-, \lambda)$, we consider the following Lyapunov function

$$V(\mu) = ||\Phi(\mu)||_2^2 + \frac{1}{2}||\mu - \mu^*||_2^2,$$

where $\Phi(y) = \begin{bmatrix} A(r, y, \gamma) \\ B(r, y, \gamma, \beta_+, \beta_-, \lambda) \\ C(r, y, \gamma) \\ D(y, \beta_+) \\ E(y, \beta_-) \\ F(y, \lambda) \end{bmatrix}$. Similar to the analysis of Theorem 46, the Jacobian matrix $\nabla\Phi$ is negative semidefinite. We have $\frac{dV(\mu)}{dt} = (\frac{d\Phi}{dt})^T\Phi + \Phi^T\frac{d\Phi}{dt} + (\mu - \mu^*)^T\frac{d\mu}{dt}$. Observe that $\frac{d\Phi}{dt} = \frac{d\Phi}{d\mu}\frac{d\mu}{dt} = \nabla\Phi(\mu)\Phi(\mu)$, we write then

$$\frac{dV(\mu)}{dt} = \underbrace{\Phi^T(\nabla\Phi(\mu)^T + \nabla\Phi(\mu))\Phi}_{\leq 0 \text{ since } \nabla\Phi \text{ is negative semidfinite}} + \underbrace{(\mu - \mu^*)^T(\Phi(\mu) - \Phi(\mu^*))}_{\leq 0 \text{ by Lemma 66}}.$$

There follows that $\frac{dV(\mu)}{dt} \leq 0$, thus $V(\mu)$ is a global Lyapunov function for the dynamical system (DNN) and (DNN) is stable in the sense of Lyapunov.
According to LaSalle's invariance principle, the trajectories $\mu(t)$ of system (DNN) converge to the largest invariant set $\mathcal{M} = \left\{\mu \mid \frac{dV(\mu)}{dt} = 0\right\}$. It is easy to see that $\frac{d\mu}{dt} = 0$ if and only if $\frac{dV}{dt} = 0$, therefore, the proposed neural network converges globally to the solution set of the problem (5.5). □

5.3.3 . Numerical experiments

To assess the performance of our dynamical neural network, we employed the multi-dimensional shape optimization problem with joint chance constraints from [89].

$$\begin{aligned} &\min_{x \in \mathbb{R}^M_{++}} \prod_{i=1}^{m} x_i^{-1}, \\ &\text{s.t} \quad \mathbb{P}\left(\sum_{j=1}^{m-1}\left(\frac{m-1}{A_{wall\,j}} x_1 \prod_{i=1, i\neq j}^{m} x_i\right), \frac{1}{A_{floor}} \prod_{j=2}^{m} x_j \leq 1\right) \geq 1-\epsilon, \\ &\quad \frac{1}{\gamma_{i,j}} x_i x_j^{-1} \leq 1, 1 \leq i \neq j \leq m. \end{aligned} \tag{5.14}$$

In our numerical experiments, we fixed the following parameters: $m = 10$, $\frac{1}{\gamma_{i,j}} = 0.5$, $\epsilon_{wall} = 0.15$, $\epsilon_{floor} = 0.15$ and $\epsilon = 0.15$. The inverse of floor's area ($\frac{1}{A_{floor}}$) and the inverse of wall area ($\frac{1}{A_{wall_j}}$) for each $j = 1, ..., m$ were considered as dependent variables. The mean value of $\frac{1}{A_{floor}}$ was set to $1.0/20.0$ and the mean values of $\frac{1}{A_{wall_j}}$, $j = 1, ..., m$ were randomly selected from the range $[1.0/60.0, 1.0/40.0]$. The value of the coefficients of the covariance matrix is set to 0.01. We first compare the results of our approach with those of the sequential convex approximation algorithm from [75] for $\theta = 10.0$ (i.e, high dependency). We test the robustness of the different approaches by creating 100 random samples of the variables $\frac{1}{A_{wall_j}}$ and $\frac{1}{A_{floor}}$ using the same mentioned moments. We then examine if the solutions from the three methods meet the constraints of (5.14) for all 100 cases. If the solutions are not viable for a particular case, it is referred to as a violated scenario (VS). The numerical results are displayed in Table 5.3. Column one gives the number of variables m. Columns two and three show the objective value and the number of VS obtained through our neural network, respectively. Columns four and five present the objective value and the number of VS obtained using the sequential algorithm. The sixth column gives the gap between the two objective values, calculated as follows: GAP = $\frac{(\text{Obj valueSA} - \text{Obj valueNN})}{\text{Obj valueSA}} \times 100$, where Obj valueNN and Obj value$_{\text{SA}}$ represent the objective values obtained by the neural network and sequential algorithm, respectively. Table 5.3 indicates that our neural network outperforms the sequential algorithm, as the upper bounds obtained are better and fewer scenarios are violated.

We solve (5.14) for different values of m and different values of θ; the dependence parameter of Gumbel-Hougaard Copula. Tables 5.4, 5.5, 5.6 show the obtained results

m	Neural Network	VS	Sequential Algorithm	VS	GAP
3	0.283	5	0.334	8	15.3
7	0.399	0	0.444	9	10.1
10	0.601	1	0.667	9	9.9
15	0.253	3	0.277	6	8.6
20	0.663	4	0.685	5	3.2

Table 5.3: Neural network vs. the sequential algorithm for normal distribution

m	$\theta = 1$ (independent constraints)		$\theta = 2$		$\theta = 10$	
	Obj value	VS	Obj value	VS	Obj value	VS
10	2.78	11	1.97	14	1.06	21
15	13.91	11	10.46	22	6.19	31
20	1.07	8	0.85	9	0.60	10
30	3.13	1	2.50	7	1.75	9

Table 5.4: Results for different values of m for normal distribution

of the normal distribution, Laplace distribution and the logistic distribution, respectively. For the three tables, column one gives the number of variables m. Columns two and three give the objective value and the number of VS when $\theta = 1$, respectively. Columns four and five show the objective value and the number of VS when $\theta = 2$, respectively. Finally, columns six and seven present the objective value and the number of VS when $\theta = 10$, respectively. We observe that for the different distributions, the objective value decreases as the dependency between the random variables becomes higher. The last observation is coherent since the joint chance constraint becomes less restrictive as θ increases. Nevertheless, the number of violated scenarios increases for the problems with high dependency.

5.4 . Linear programs as a particular case

This section was published in Results in Control and Optimization Journal [131].
We study now the following stochastic linear programming problem

$$\begin{aligned} \min\ & c^T x, \\ \text{s.t.}\ & \mathbb{P}(\mathsf{T}_k x \leq \mathsf{D}_k, k = 1, .., K) \geq 1 - \alpha, \\ & x \geq 0, \end{aligned} \tag{5.15}$$

where T_k, $k = 1, .., K$ are random vector rows, D_k, $k = 1, .., K$ are constants and $1 - \alpha$ is a given confidence level. In our study, the dependence among the random rows Tk, $k = 1, .., K$, is modeled using the Gumbel-Hougaard copula with a parameter θ. Each row vector Tk is characterized by a known mean vector $\mu_k = (\mu_{k_1}, ..., \mu_{k_n})^T$ and a covariance

m	$\theta = 1$ (independent constraints)		$\theta = 2$		$\theta = 10$	
	Obj value	VS	Obj value	VS	Obj value	VS
10	2.30	24	1.34	30	0.86	36
15	4.22	19	2.47	26	1.53	31
20	5.19	10	3.79	15	2.33	18
30	11.49	2	6.83	5	4.21	8

Table 5.5: Results for different values of m for Laplace distribution

m	$\theta = 1$ (independent constraints)		$\theta = 2$		$\theta = 10$	
	Obj value	VS	Obj value	VS	Obj value	VS
10	0.75	1	0.28	4	0.21	7
15	2.21	4	1.35	5	0.75	6
20	3.30	5	2.37	7	1.38	9
30	4.98	1	3.56	5	2.05	9

Table 5.6: Results for different values of m for Logistic distribution

matrix Σ_k.

5.4.1 . Deterministic formulation equivalent

Problem (5.15) can be equivalently formulated as [27]

$$\begin{aligned} \min\ & c^T x, \\ \text{s.t.}\ & \mu_k^T x + \phi^{-1}((1-\alpha)^{y_k \frac{1}{\theta}})\sqrt{x^T \Sigma_k x} \leq \mathsf{D}_k, k = 1, .., K, \\ & \sum_{k=1}^{K} y_k = 1, \\ & y_k \geq 0, k = 1, .., K, \\ & x \geq 0, \end{aligned} \tag{5.16}$$

where ϕ is one-dimensional standard normal distribution.

The convexity of the feasible set of problem (5.16) is guaranteed under certain conditions, as stated in the following Theorem. The detailed proof of this Theorem can be found in the work of Cheng et al. [27].

Theorem 41. [27]. If $1 - \alpha \geq max(\phi(u^*, v^*))$ where

$$u^* = \mathsf{max}_k \frac{r_k + 1}{r_k - 1} (\lambda_{min}^k)^{\frac{-1}{2}} ||\mu_k||$$

and

$$v^* = \mathsf{max}_k \frac{r_k + 1}{\sqrt{(r_k + 1) + (\frac{1-\theta}{2pln(p)} f_\phi(\sqrt{r_k + 1}))^2} + \frac{1-\theta}{2pln(p)} f_\phi(\sqrt{r_k + 1})},$$

for some $r_k \geq 1$ with λ_{min} is the smallest eigenvalue of Σ_k, ϕ is the standard normal distribution and f_ϕ is the associated density, then the feasible set of (8)

$$X(\alpha) = \left\{ x \geq 0 \mid \forall k = 1, .., K, \exists y_k \geq 0 : \mu_k^T x + \phi^{-1}((1-\alpha)^{y_k \frac{1}{\theta}})\sqrt{x^T \Sigma x} \leq D_k, \sum_{k=1}^{K} y_k = 1 \right\}$$

is convex.

Corollary 42. By Theorem 41, problem (5.16) is biconvex.

Let $f(x) = c^T x$, $g_{\theta_k}(x, y) = \mu_k^T x + \phi^{-1}((1-\alpha)^{y_k \frac{1}{\theta}})\sqrt{x^T \Sigma_k x} - \mathsf{D}_k$, $l(y) = (\sum_{k=1}^{K} y_k - 1, 1 - \sum_{k=1}^{K} y_k)^T$, $h_k(y) = -y_k$ and $w(x) = -x$. We write problem (5.15) as

$$\begin{aligned} \min \ & f(x), \\ \text{s.t.} \ & g_{\theta k}(x, y) \leq 0, k = 1, .., K, \\ & l(y) \leq 0, \\ & h_k(y) \leq 0, k = 1, .., K, \\ & w(x) \leq 0, \end{aligned} \tag{5.17}$$

Theorem 43. Let $\mathcal{U}$ the feasible set of (5.16), let $(x^*, y^*) \in \mathcal{U}$. If there are $\mu^{(1)}$, $\mu^{(2)}$, γ, λ, ω such that

$$\nabla_x f(x^*) + {\mu^{(1)}}^T \nabla_x g_\theta(x^*, y^*) + \omega^T \nabla_x w(x^*) = 0, \tag{5.18}$$

$$\mu^{(1)} \geq 0, {\mu^{(1)}}^T g_\theta(x^*, y^*) = 0, \omega \geq 0, \omega^T w(x^*) = 0, \tag{5.19}$$

$${\mu^{(2)}}^T \nabla_y g_\theta(x^*, y^*) + \gamma^T \nabla_y l(y^*) + \lambda^T \nabla_y h(y^*) = 0, \tag{5.20}$$

$$\mu^{(2)} \geq 0, {\mu^{(2)}}^T g_\theta(x^*, y^*) = 0, \gamma \geq 0, \gamma^T l(y*) = 0, \tag{5.21}$$

$$\lambda \geq 0, \lambda^T h(y^*) = 0, \tag{5.22}$$

with $g_\theta(x, y) = (g_{\theta_1}(x, y), \ldots, g_{\theta_K}(x, y))^T$ and $h(y) = (h_1(y), \ldots, h_K(y))^T$, then (x^*, y^*) is a partial KKT point of (5.17).

As stated in the previous Section, if (5.17) is satisfied with partial Slater constraints qualificationt (x^*, y^*), then (x^*, y^*) is a partial optimum of (5.17) if and only if (x^*, y^*) is a partial KKT point of (5.17). Moreover, if $\mu^{(1)} = \mu^{(2)}$ then (x^*, y^*) is a KKT point of (5.17).

5.4.2 . The dynamical neural network

The neural network to solve (5.17) is driven by the following differential system

$$\frac{dx}{dt} = -(\nabla_x f(x) + \nabla_x g_\theta(x,y)^T(\mu + g_\theta(x,y))_+ + \nabla_x w(x)^T(\omega + w(x))_+), \tag{5.23}$$

$$\frac{dy}{dt} = -(\nabla_y g_\theta(x,y)^T(\mu + g_\theta(x,y))_+ + \nabla_y l(y)^T(\gamma + l(y))_+ + \nabla_y h(y)^T(\lambda + h(y))_+), \tag{5.24}$$

$$\frac{d\mu}{dt} = (\mu + g_\theta(x,y))_+ - \mu, \tag{5.25}$$

$$\frac{d\gamma}{dt} = (\gamma + l(y))_+ - \gamma, \tag{5.26}$$

$$\frac{d\lambda}{dt} = (\lambda + h(y))_+ - \lambda, \tag{5.27}$$

$$\frac{d\omega}{dt} = (\omega + w(x))_+ - \omega. \tag{5.28}$$

We rewrite the dynamical system as $\begin{cases} \frac{dz}{dt} = \Phi(z) \\ z(t_0) = z_0 \end{cases}$,
where $z = (x, y, \mu, \gamma, \lambda, \omega)$ and

$$\Phi(z) = \begin{bmatrix} -(\nabla_x f(x) + \nabla_x g_\theta(x,y)^T(\mu + g_\theta(x,y))_+ + \nabla_x w(x)^T(\omega + w(x))_+) \\ -(\nabla_y g_\theta(x,y)^T(\mu + g_\theta(x,y))_+ + \nabla_y l(y)^T(\gamma + l(y))_+ + \nabla_y h(y)^T(\lambda + h(y))_+) \\ (\mu + g_\theta(x,y))_+ - \mu \\ (\gamma + l(y))_+ - \gamma \\ (\lambda + h(y))_+ - \lambda \\ (\omega + w(x))_+ - \omega \end{bmatrix}.$$

The neural network system given by equations (5.23)-(5.28) can be implemented as an autonomous differential system using a generalized circuit. Figure 5.3 illustrates a generalized circuit implementation of the neural network. To begin the computation, we initialize the values of the variables x, y, μ, γ, λ, and ω with the initial values x_0, y_0, μ_0, γ_0, λ_0, and ω_0, respectively. We then solve equations (5.23)-(5.28) iteratively until convergence is achieved.

The following Theorem establishes the equivalence between an equilibrium point of the neural network system (5.23)-(5.28) and a Karush-Kuhn-Tucker (KKT) point of the optimization problem (5.16).

Theorem 44. Let $(x^*, y^*, \mu^*, \gamma^*, \lambda^*, \omega^*)$ an equilibrium point of (5.23)-(5.28), then (x^*, y^*) is a partial KKT point of (5.16) with $\mu^{(1)} = \mu^{(2)}$. If (x^*, y^*) is a partial optimum of (5.16), then there exist $(\mu^*, \gamma^*, \lambda^*, \omega^*)$ such that $(x^*, y^*, \mu^*, \gamma^*, \lambda^*, \omega^*)$ is an equilibrium point of (5.23)-(5.28).

Proof. Let $(x^*, y^*, \mu^*, \gamma^*, \lambda^*, \omega^*)$ be an equilibrium point of (5.23)-(5.28), we have then $\frac{dx^*}{dt} =$

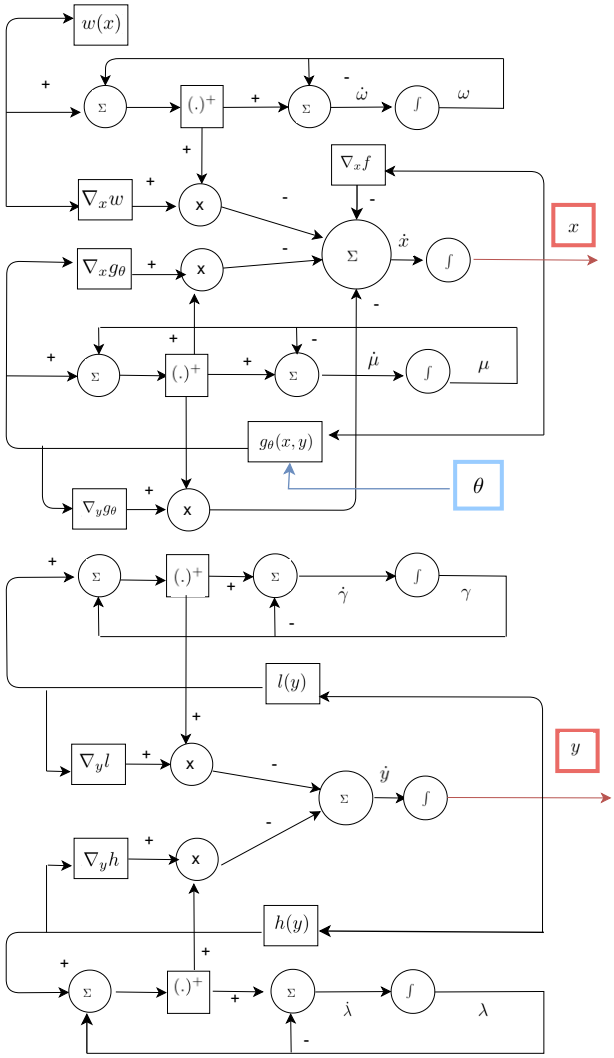

Figure 5.3: A block diagram for the neural network (5.23)-(5.28)

0, $\frac{dy^*}{dt} = 0$, $\frac{d\mu^*}{dt} = 0$, $\frac{d\gamma^*}{dt} = 0$, $\frac{d\lambda^*}{dt} = 0$ and $\frac{d\omega^*}{dt} = 0$. There follows that

$$\nabla_x f(x^*) + \nabla_x g_\theta(x^*, y^*)^T(\mu^* + g_\theta(x^*, y^*))_+ + \nabla_x w(x^*)^T(\omega^* + w(x^*))_+ = 0,$$
$$\nabla_y g_\theta(x^*, y^*)^T(\mu + g_\theta(x^*, y^*))_+ + \nabla_y l(y^*)^T(\gamma^* + l(y^*))_+ + \nabla_y h(y^*)^T(\lambda^* + h(y^*))_+ = 0,$$
$$(\mu^* + g_\theta(x^*, y^*))_+ - \mu^* = 0,$$
$$(\gamma^* + l(y^*))_+ - \gamma^* = 0,$$
$$(\lambda^* + h(y^*))_+ - \lambda^* = 0,$$
$$(\omega^* + w(x^*))_+ - \omega^* = 0.$$

We observe that $(\mu^* + g_\theta(x^*, y^*))_+ - \mu^* = 0$ if and only if $\mu^* \geq 0$, $g_\theta(x^*, y^*) \leq 0$ and

${\mu^*}^T g_\theta(x^*, y^*) = 0$. The same observation applies for $(\gamma^* + l(y^*))_+ - \gamma^* = 0$, $(\lambda^* + h(y^*))_+ - \lambda^* = 0$ and $(\omega^* + w(x^*))_+ - \omega^* = 0$. By substitution, we find the KKT system (5.18)-(5.22). The proof of the converse is straightforward and the conclusion follows. □

Theorem 45. For any initial point $(x(t_0), y(t_0), \mu(t_0), \gamma(t_0), \lambda(t_0), \omega(t_0))$, there exists an unique continuous solution $(x(t), y(t), \mu(t), \gamma(t), \lambda(t), \omega(t))$ for (5.23)-(5.28).

Proof. Since $\nabla_x f(x)$, $\nabla_x g_\theta(x, y)$, $\nabla_y g_\theta(x, y)$, $\nabla_x w(x)$, $\nabla_y h(y)$ and $\nabla_y l(y)$ are continuously differentiable on open sets, then all the second terms of the differential equations (5.23)-(5.28) are locally Lipschitz continuous. According to the local existence of ordinary differential equations also known as *Picard–Lindelöf Theorem* [77], the neural network (5.23)-(5.28) has a unique continuous solution $(x(t), y(t), \mu(t), \gamma(t), \lambda(t), \omega(t))$. □

To show the stability of the neural network (5.23)-(5.28), we need to show the semidefiniteness of the Jacobian matrix $\nabla\Phi(z)$.

Theorem 46. The Jacobian matrix $\nabla\Phi(z)$ is negative semidefinite.

Proof. We consider the dynamical neural network (5.23)-(5.28). We assume that there exist $0 \leq p \leq K$, $0 \leq q \leq 2$, $0 \leq s \leq K$ and $0 \leq t \leq n$ such that $(\mu + g_\theta)_+ = (\mu_1 + g_{\theta 1}(x, y), \mu_2 + g_{\theta 2}(x, y), ..., \mu_p + g_{\theta p}(x, y), \underbrace{0, ..., 0}_{K-p})$,

$(\gamma + l)_+ = (\gamma_1 + l_1(y), \gamma_2 + l_2(y), ..., \gamma_q + l_q(y), \underbrace{0, ..., 0}_{2-q})$,

$(\lambda + h)_+ = (\lambda_1 + h_1(y), \lambda_2 + h_2(y), ..., \lambda_s + h_s(y), \underbrace{0, ..., 0}_{K-s})$,

$(\omega + w)_+ = (\omega_1 + w_1(x), \omega_2 + w_2(x), ..., \omega_t + w_t(x), \underbrace{0, ..., 0}_{n-t})$.

We write the Jacobian matrix of ϕ as follows $\nabla\Phi(z) = \begin{bmatrix} A_1 & A_2 & A_3 & 0 & 0 & A_6 \\ B_1 & B_2 & B_3 & B_4 & B_5 & 0 \\ C_1 & C_2 & C_3 & 0 & 0 & 0 \\ 0 & D_2 & 0 & D_4 & 0 & 0 \\ 0 & E_2 & 0 & 0 & E_5 & 0 \\ F_1 & 0 & 0 & 0 & 0 & F_6 \end{bmatrix}$,

where

$$A_1 = -(\nabla_x^2 f(x) + \sum_{i=1}^{p}((\mu_i + g_{\theta i})\nabla_x^2 g_{\theta_i}^p(x, y)) + \nabla_x g_\theta{}^p(x, y)^T \nabla_x g_\theta{}^p(x, y) + \sum_{i=1}^{t}((\omega_i + w_i)\nabla_x^2 w_i^t(x)) + \nabla_x w^t(x)^T \nabla_x w^t(x)),$$

$$A_2 = -(\sum_{i=1}^{p}((\mu_i + g_{\theta i})\nabla_y \nabla_x g_{\theta_i}^p(x, y)) + \nabla_y g_\theta{}^p(x, y)^T \nabla_x g_\theta{}^p(x, y)),$$

$$B_1 = -(\sum_{i=1}^{p}((\mu_i + g_{\theta i})\nabla_x \nabla_y g_{\theta}{}_i^p(x,y)) + \nabla_x g_\theta{}^p(x,y)^T \nabla_y g_\theta{}^p(x,y)),$$

$$B_2 = -(\sum_{i=1}^{p}((\mu_i + g_{\theta i})\nabla_y^2 g_{\theta}{}_i^p(x,y)) + \nabla_y g_\theta{}^p(x,y)^T \nabla_y g_\theta{}^p(x,y) + \sum_{i=1}^{q}((\gamma_i + l)\nabla_y^2 l_i^q(y))$$
$$+ \nabla_y l^q(y)^T \nabla_y l^q(y) + \sum_{i=1}^{s}((\lambda_i + h)\nabla_y^2 h_i^s(y)) + \nabla_y h^s(y)^T \nabla_y h^s(y)),$$

$$A_3 = -\nabla_x g_\theta^p(x,y)^T, \quad A_6 = -\nabla_x w^t(x)^T, \quad B_3 = -\nabla_y g_\theta^p(x,y)^T$$
$$B_4 = -\nabla_y l^q(y)^T, \quad B_5 = -\nabla_y h^s(y)^T, \quad C_1 = \nabla_x g_\theta^p(x,y),$$
$$C_2 = \nabla_y g_\theta^p(x,y), \quad D_2 = \nabla_y l^q(y), \quad E_2 = \nabla_y h^s(y),$$
$$F_1 = \nabla_x w^t(x),$$

$$C_3 = S_p = -\begin{bmatrix} O_{p\times p} & O_{p\times(K-p)} \\ O_{(K-p)\times p} & I_{(K-p)\times(K-p)} \end{bmatrix}, \quad D_4 = S_q = -\begin{bmatrix} O_{q\times q} & O_{q\times(2-q)} \\ O_{(2-q)\times q} & I_{(2-q)\times(2-q)} \end{bmatrix},$$
$$E_5 = S_s = -\begin{bmatrix} O_{s\times s} & O_{s\times(K-s)} \\ O_{(K-s)\times q} & I_{(K-s)\times(K-s)} \end{bmatrix} \quad \text{and} \quad F_6 = S_t = -\begin{bmatrix} O_{t\times t} & O_{t\times(n-t)} \\ O_{(n-t)\times t} & I_{(n-t)\times(n-t)} \end{bmatrix}.$$

The Jacobian matrix becomes $\nabla\Phi(z) = \begin{bmatrix} A_1 & B_1^T & A_3 & 0 & 0 & A_6 \\ B_1 & B_2 & B_3 & B_4 & B_5 & 0 \\ -A_3^T & -B_3^T & S_p & 0 & 0 & 0 \\ 0 & -B_4^T & 0 & S_q & 0 & 0 \\ 0 & -B_5^T & 0 & 0 & S_s & 0 \\ -A_6^T & 0 & 0 & 0 & 0 & S_t \end{bmatrix}$.

Let $A = \begin{bmatrix} A_1 & B_1^T \\ B_1 & B_2 \end{bmatrix}$, $B = \begin{bmatrix} A_3 & 0 & 0 & A_6 \\ B_3 & B_4 & B_5 & 0 \end{bmatrix}$ and $S = \begin{bmatrix} S_p & 0 & 0 & 0 \\ 0 & S_q & 0 & 0 \\ 0 & 0 & S_s & 0 \\ 0 & 0 & 0 & S_t \end{bmatrix}$, we have then

$\nabla\Phi(z) = \begin{bmatrix} A & B \\ -B^T & S \end{bmatrix}$.

It is clear that S is negative semidefinite. Furthermore, since g_θ is biconvex, l, h and w are convex then $\nabla_x^2 g_{\theta}{}_i^p(x,y)$, $\nabla_x^2 w_i^p(x)$, $\nabla_y^2 g_{\theta}{}_i^p(x,y)$, $\nabla_y^2 l_i^q(y)$ and $\nabla_y^2 h_i^s(y)$ are positive semidefinite. It is easily proved that $\nabla_x g_\theta{}^p(x,y)^T \nabla_x g_\theta{}^p(x,y)$, $\nabla_x w^t(x)^T \nabla_x w^t(x)$, $\nabla_y g_\theta{}^p(x,y)^T \nabla_y g_\theta{}^p(x,y)$, $\nabla_y l^q(y)^T \nabla_y l^q(y)$ and $\nabla_y h^s(y)^T \nabla_y h^s(y)$ are positive semidefinite. There follows that A_1 and B_2 are negative semidefinite; hence, A is negative semidefinite. Since A and S are negative semidefinite, we conclude that $\nabla\Phi$ is negative semidefinite [48]. □

Theorem 47. The neural network (5.23)-(5.28) is stable and converges to a partial optimum of (5.16).

Proof. Let $z^* = (x^*, y^*, \mu^*, \gamma^*, \lambda^*, \omega^*)$ an equilibrium point for the neural network (5.23)-(5.28). We consider the following Lyapunov function

$$E_1(z) = \|\Phi(z)\|^2 + \frac{1}{2}\|z - z^*\|^2$$

we have that $\frac{d\Phi}{dt} = \frac{\nabla\Phi}{\nabla z}\frac{dz}{dt} = \nabla\Phi(z)\Phi(z)$, then $\frac{dE_1(z(t))}{dt} = (\frac{d\Phi}{dt})^T\Phi + \Phi^T\frac{d\Phi}{dt} + (z-z^*)^T\frac{dz}{dt} = \Phi^T(\nabla\Phi(z)^T + \nabla\Phi(z))\Phi + (z-z^*)^T\Phi(z)$.

By Theorem 46, we have $\Phi^T(\nabla\Phi(z)^T + \nabla\Phi(z))\Phi \leq 0$. By Theorem 46 and Lemma 24, we have $(z-z^*)^T\Phi(z) = (z-z^*)^T(\Phi(z) - \Phi(z^*)) \leq 0$ then $\frac{dE_1(z(t))}{dt} \leq 0$. $E_1(z)$ is positive and $\frac{dE_1(z(t))}{dt} \leq 0$ then the neural network (5.23)-(5.28) is globally stable in the sense of Lyapunov.

As $E_1(z) \geq \frac{1}{2}\|z-z^*\|^2$, then there exists a convergent subsequence $(z(t_k)_{k\geq 0})$ where $t_k \longrightarrow \infty$ when $k \longrightarrow \infty$, such that $lim_{k\longrightarrow\infty} z(t_k) = \hat{z}$, where $\hat{z}$ satisfies $\frac{dE_1(z(t))}{dt} = 0$. Notice that $\hat{z}$ is a w-limit point of $\{z(t)\}_{t\geq t_0}$. We have by LaSalle's invariance principle [128] that the neural network converges to the largest invariant set contained in M which is defined by $M = \{z(t)|\frac{dE_1(z)}{dt} = 0\}$. We observe that $(\frac{dx}{dt} = 0, \frac{dy}{dt} = 0, \frac{d\mu}{dt} = 0, \frac{d\gamma}{dt} = 0, \frac{d\lambda}{dt} = 0$ and $\frac{d\omega}{dt} = 0) \Leftrightarrow \frac{dE_1(z)}{dt} = 0$, therefore, $\hat{z}$ is an equilibrium point for the neural network (5.23)-(5.28).
We now define a new Lyapunov function

$$E_2(z) = \|\Phi(z)\|^2 + \frac{1}{2}\|z-\hat{z}\|^2$$

$E_2(z)$ is continuously differentiable, $E_2(\hat{z}) = 0$ and $lim_{k\longrightarrow\infty} z(t_k) = \hat{z}$ then $lim_{t\longrightarrow\infty} E_2(z(t)) = E_2(\hat{z})$. We have also $\frac{dE_2(z)}{dt} \leq 0$, then $\frac{1}{2}\|z-\hat{z}\|^2 \leq E_2(z(t))$. Hence, $lim_{t\longrightarrow\infty}\|z-\hat{z}\| = 0$ and $lim_{t\longrightarrow\infty} z(t) = \hat{z}$. Therefore, the neural network (5.23)-(5.28)) is convergent in the sense of Lyupanov to an equilibrium point $\hat{z} = (\hat{x}, \hat{y}, \hat{\mu}, \hat{\gamma}, \hat{\lambda}, \hat{\omega})$ where $(\hat{x}, \hat{y})$ is a partial optimum of problem (5.16). □

5.4.3 . Numerical experiments

To assess the performance of our approach, we conducted computational tests on a standard profit maximization problem. We compared the obtained results with a tangent approximation method proposed by Cheng et al. [25].

A manufacturing firm operates three machines to produce three different products. The manufacturing time for each unit of the products follows a normal distribution. Table 5.7 provides the mean and standard deviation values for the manufacturing time of each product, as well as the daily capacity of the machines. The first column lists the machine names, while columns 2 to 7 display the mean and standard deviation values for each product. The machine capacities are listed in column 8.

Our objective is to find the optimal daily production quantities for each product, taking into account the available machining times. The profit per unit for product 1, 2, and 3 is given by $p_1 = 60$, $p_2 = 65$, and $p_3 = 62$, respectively. We can formulate the minimization problem as follows

$$\begin{aligned} \min\ & -60x_1 - 65x_2 - 62x_3, && (5.29)\\ \text{s.t.}\ & \mathbb{P}(t_{11}x_1 + t_{12}x_2 + t_{13}x_3 \leq 680, \\ & t_{21}x_1 + t_{22}x_2 + t_{23}x_3 \leq 660, \\ & t_{31}x_1 + t_{32}x_2 + t_{33}x_3 \leq 710) \geq 0.95, && (5.30)\\ & x_1, x_2, x_3 \geq 0. && (5.31) \end{aligned}$$

Machine	Time per unit (min)						Machine capacity (min/day)
	Product 1		Product 2		Product 3		
	mean	standard deviation	mean	standard deviation	mean	standard deviation	
Machine 1	$\bar{t_{11}} = 12$	$\sigma_{11} = 2.3$	$\bar{t_{12}} = 12$	$\sigma_{12} = 2.0$	$\bar{t_{13}} = 12$	$\sigma_{13} = 2.0$	$C_1 = 680$
Machine 2	$\bar{t_{21}} = 12$	$\sigma_{21} = 2.9$	$\bar{t_{22}} = 12$	$\sigma_{22} = 2.3$	$\bar{t_{23}} = 13$	$\sigma_{23} = 2.0$	$C_2 = 660$
Machine 3	$\bar{t_{31}} = 14$	$\sigma_{31} = 2.9$	$\bar{t_{32}} = 14$	$\sigma_{32} = 2.9$	$\bar{t_{33}} = 12$	$\sigma_{33} = 2.9$	$C_3 = 720$

Table 5.7: Data

The resulting deterministic equivalent problem of (5.29)- (5.31) is given as follows

$$
\begin{aligned}
\min\ & -60x_1 - 65x_2 - 62x_3, \\
\text{s.t.}\ & \bar{t}_{11}x_1 + \bar{t}_{12}x_2 + \bar{t}_{13}x_3 + \phi^{-1}((0.95)^{y_1 \frac{1}{\theta}})\sqrt{\sigma_{11}^2 x_1^2 + \sigma_{12}^2 x_2^2 + \sigma_{13}^2 x_3^2} \leq 680, \\
& \bar{t}_{21}x_1 + \bar{t}_{22}x_2 + \bar{t}_{23}x_3 + \phi^{-1}((0.95)^{y_2 \frac{1}{\theta}})\sqrt{\sigma_{21}^2 x_1^2 + \sigma_{22}^2 x_2^2 + \sigma_{23}^2 x_3^2} \leq 660, \\
& \bar{t}_{31}x_1 + \bar{t}_{32}x_2 + \bar{t}_{33}x_3 + \phi^{-1}((0.95)^{y_3 \frac{1}{\theta}})\sqrt{\sigma_{31}^2 x_1^2 + \sigma_{32}^2 x_2^2 + \sigma_{33}^2 x_3^2} \leq 710, \\
& y_1 + y_2 + y_3 = 1, \\
& x_1, x_2, x_3, y_1, y_2, y_3 \geq 0.
\end{aligned}
$$

We consider that $[t_{11}, t_{12}, t_{13}]^T$, $[t_{21}, t_{22}, t_{23}]^T$ and $[t_{31}, t_{32}, t_{33}]^T$ are dependent and the dependence is driven by Gumbel-Hougaard copula. We solve problem (5.29)-(5.31) using the proposed neural network for $\theta = 2$ and $\theta = 10$. We compare the results obtained with the dynamical neural network with those obtained using the piecewise tangent approximation with 20 tangent points [25]. In order to evaluate the robustness of our approach, we generate a set of 100 instances of the stochastic vectors $[t_{11}, t_{12}, t_{13}]^T$, $[t_{21}, t_{22}, t_{23}]^T$ and $[t_{31}, t_{32}, t_{33}]^T$ with the data in Table 5.7 and the function *numpy.random.multivariate_normal*. Then, we check whether the two solutions are feasible for the constraints (5.29)-(5.31) for the 100 scenarios. If the solutions are not feasible for a given scenario, we call such a scenario a violated scenario denoted (VS).

$\theta = 1$ (independent constraints)			$\theta = 2$			$\theta = 10$		
Solution	VS	CPU Time	Solution	VS	CPU Time	Solution	VS	CPU Time
-2929.76	4	7.65	-2961.40	5	10.44	-2986.594	6	31.03

Table 5.8: Results of problem (5.29)-(5.31) for different values of θ

We observe that the neural network converges for the different values of θ and the number of violated scenarios doesn't exceed 6 which proves the robustness of the approach as shown by Figure 5.4.

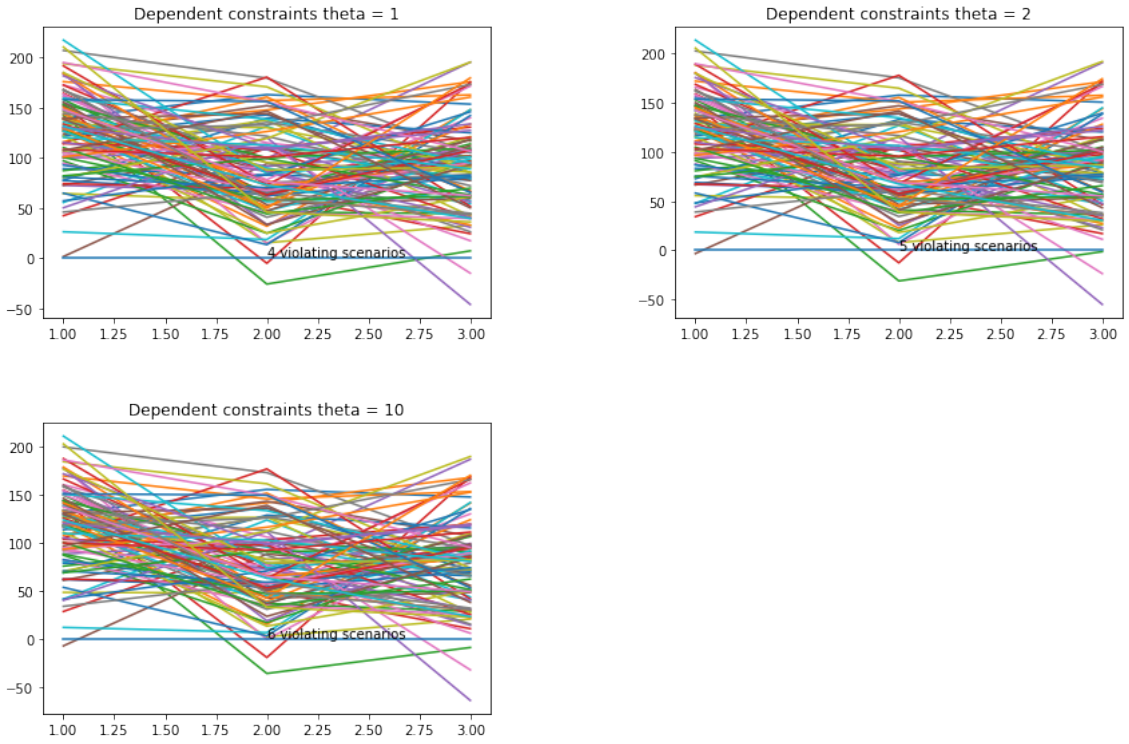

Figure 5.4: Number of VS for different values of θ

In order to compare the robustness for different values of θ, we solve problem (5.29)-(5.31) for different values of the confidence parameter α. We recapitulate the obtained results for different values of α in Table 5.9.

We observe that for all the values of α the dynamical neural network converges using the three values of θ. The solutions are all robust as they don't exceed the number of permitted violations. We notice that the solutions are more conservative when α increases.

We fix $\alpha = 0.10$ and we modify the number of machines k and the number of products n. We compare the results obtained using the neural network with the tangent approximation [27]. We use 20 tangent points for the tangent approximation and we recapitulate the results in Table 5.10. We notice that the gaps remain tight i.e they do not exceed 0.15% in the same way as in the independent case.

α	$\theta = 1$ (independent constraints)			$\theta = 2$			$\theta = 10$		
	Obj value	VS	CPU Time	Obj value	VS	CPU Time	Obj value	VS	CPU Time
0.10	-2803.75	1	9.50	-2856.78	1	9.46	-2897.36	4	2.60
0.15	-2863.54	2	9.89	-2926.52	5	9.19	-2976.50	8	2.99
0.20	-2911.68	2	9.42	-2983.73	9	9.70	-3041.72	11	3.22
0.25	-2953.43	9	10.86	-3034.27	15	10.33	-3100.25	24	3.83

Table 5.9: Results for different values of α

n	k	$\theta = 2$					$\theta = 10$				
		Tangent approximation	VS	Neural network	VS	GAP	Tangent approximation	VS	Neural network	VS	GAP
5	3	-2422.18	0	-2420.41	0	0.07%	-2425.46	1	-2425.16	1	0.01%
7	5	-2524.58	5	-2523.07	5	0.05%	-2545.96	10	-2545.78	10	0.007%
10	5	-2638.22	3	-2634.13	2	0.15%	-2675.62	9	-2674.92	9	0.02%
15	10	-2839.43	4	-2837.83	4	0.05%	-2867.28	7	-2867.05	7	0.08%

Table 5.10: Results for different data sizes with $\alpha = 0.10$

6 - Distributionally Robust Optimization

6.1 . Introduction

Uncertainty in parameters is a common characteristic of real-world decision problems in engineering and management. This uncertainty can arise from factors such as limited data observability, noisy measurements, implementation variations, and prediction errors. To address this uncertainty, stochastic optimization, (SO) and robust optimization (RO) frameworks have been traditionally employed in decision-making processes.

While SO assumes that the decision-maker possesses complete knowledge about the underlying uncertainty through a known probability distribution and minimizes a cost function based on this distribution, RO assumes that the decision-maker lacks detailed distributional knowledge about the uncertainty, except for its support. In RO, the goal is to minimize the worst-case cost over an uncertainty set, as studied in the works of El Ghaoui and Lebret [43], Ben-Tal and Nemirovski [12], and Bertsimas and Sim [14].

Distributionally Robust Optimization (DRO) is a framework used to address optimization problems in the presence of uncertain probability distributions. The goal of DRO is to find robust solutions that perform well across a range of possible distributions, rather than assuming a single fixed distribution.

Recent papers have considered the use of distributionally robust approaches in transportation network optimization problems [35], multistage distribution system planning [145], portfolio optimization problems [49, 140], planning and scheduling [121], risk measures [108], multimodal demand problems [60], appointment scheduling [148], vehicle routine problems [52] and energy and reserve dispatch [104].

6.2 . Geometric Programs

We consider the following geometric program.

$$\begin{aligned} \min_{t \in \mathbb{R}^M_{++}} \quad & \sum_{i=1}^{I_0} c_i^0 \prod_{j=1}^{M} t_j^{a_{ij}^0}, \\ \text{s.t} \quad & \sum_{i=1}^{I_k} c_i^k \prod_{j=1}^{M} t_j^{a_{ij}^k} \leq 1, k = 1, ..., K, \end{aligned} \tag{6.1}$$

where c_i^k, $i = 1, ..., I_k$, $k = 0, ..., K$ are positive constants and the exponents $a_{ij}^k, i = 1, ..., I_k, j = 1, ..., M, k = 0, 1, ..., K$ are real constants.

In this paper, we consider the case where the coefficients c_i are not known. In conse-

quence, we reformulate the optimization problem (6.1) as follows

$$\min_{t\in\mathbb{R}^M_{++}} \sup_{\mathcal{F}_0\in\mathcal{D}_0} \mathbb{E}_{\mathcal{F}_0}\left[\sum_{i=1}^{I_0} c_i^0 \prod_{j=1}^{M} t_j^{a_{ij}^0}\right], \tag{JCP}$$
$$\text{s.t} \quad \inf_{\mathcal{F}\in\mathcal{D}} \mathbb{P}_{\mathcal{F}}\left(\sum_{i=1}^{I_k} c_i^k \prod_{j=1}^{M} t_j^{a_{ij}^k} \leq 1, k = 1, ..., K\right) \geq 1-\epsilon,$$

where $\mathcal{F}_0$ is the probabilistic distribution of vector $C^0 = (c_1^0, .., c_{I_0}^0)^T$, $\mathcal{F}$ is the joint distribution for $C^1 = (c_1^1, .., c_{I_1}^1)^T$, ..., $C^k = (c_1^k, .., c_{I_k}^k)^T$, $\mathcal{D}_0$ is the uncertainty set for the probability distribution $\mathcal{F}_0$, $\mathcal{D}$ is the uncertainty set for the probability distribution $\mathcal{F}$ and $1-\epsilon$, $\epsilon \in (0, 0.5]$, is the confidence parameter for the joint constraint.

This paper examines the distributionally robust geometric programs (JCP) using two different groups sets of uncertainty. The first group focuses on uncertainties in distributions, considering both known and unknown first two order moments. The second group of uncertainty sets incorporates first order moments along with nonnegative support constraints.

6.2.1 . Uncertainty Sets with First Two Order Moments

We first consider that the mean vector of C^k, $k = 0, 1, ..., K$ lies in an ellipsoid of size $\gamma_1^k \geq 0$ with center μ_k and that the covariance matrix of C^k, $k = 0, 1, ..., K$ lies in a positive semidefinite cone of center $\Sigma_k = \left\{\sigma_{i,j}^k, i, j = 1, ..., I_k\right\}$. We define for every $k = 0, 1, ..., K$, $\mathcal{D}_k^2(\mu_k, \Sigma_k) = \left\{\mathcal{F}_k \;\middle|\; \begin{array}{c}(\mathbb{E}_{\mathcal{F}_k}[C^k] - \mu_k)^T \Sigma_k^{-1} (\mathbb{E}_{\mathcal{F}_k}[C^k] - \mu_k) \leq \gamma_1^k \\ \text{COV}_{\mathcal{F}_k}(C^k) \preceq \gamma_2^k \Sigma_k\end{array}\right\}$, where F_k is the probability distribution of C^k, $\gamma_2^k \geq 0$ and $\text{COV}_{\mathcal{F}_k}$ is a covariance operator under probability distribution $\mathcal{F}_k$ of C^k.

Based on whether the row vectors C^k, $k = 1, .., K$ are mutually independent or dependent, we have two cases.

Case (JCP) with Jointly Independent Row Vectors.

Assumption 48. We assume that $\mathcal{D} = \{\mathcal{F} | \mathcal{F} = \mathcal{F}_1 \mathcal{F}_2 ... \mathcal{F}_K\}$, where $\mathcal{F}$ is the joint distribution for mutually independent random vectors C^1 , C^2 , ..., C^K with marginals $\mathcal{F}_1$, $\mathcal{F}_2$, ..., $\mathcal{F}_K$.

Theorem 49. Given Assumption 48, (JCP) is equivalent to

$$(\text{JCP}_{ind}) \min_{t\in\mathbb{R}^M_{++}, y\in\mathbb{R}^K_+} \sum_{i=1}^{I_0}\mu_i^0\prod_{j=1}^{M}t_j^{a_{ij}^0}+\sqrt{\gamma_1^0}\sqrt{\sum_{i=1}^{I_0}\sum_{l=1}^{I_0}\sigma_{i,l}^0\prod_{j=1}^{M}t_j^{a_{ij}^0+a_{lj}^0}}, \tag{6.2}$$

$$\text{s.t} \quad \sum_{i=1}^{I_k}\mu_i^k\prod_{j=1}^{M}t_j^{a_{ij}^k}+\sqrt{\gamma_1^k}\sqrt{\sum_{i=1}^{I_k}\sum_{l=1}^{I_k}\sigma_{i,l}^k\prod_{j=1}^{M}t_j^{a_{ij}^k+a_{lj}^k}}$$

$$+\sqrt{\frac{y_k}{1-y_k}}\sqrt{\gamma_2^k}\sqrt{\sum_{i=1}^{I_k}\sum_{l=1}^{I_k}\sigma_{i,l}^k\prod_{j=1}^{M}t_j^{a_{ij}^k+a_{lj}^k}}\leq 1,\ k=1,...,K, \tag{6.3}$$

$$\prod_{k=1}^{K}y_k\geq 1-\epsilon,\ 0<y_k\leq 1\,,\ k=1,...,K. \tag{6.4}$$

Proof. As the row vectors C^k, $k=1,...,K$ are mutually independent, (JCP) is written equivalently by introducing K nonegative auxiliary variables y_k as [132].

$$\min_{t\in\mathbb{R}^M_{++}}\sup_{\mathcal{F}_0\in\mathcal{D}_0}\mathbb{E}_{\mathcal{F}_0}\left[\sum_{i=1}^{I_0}c_i^0\prod_{j=1}^{M}t_j^{a_{ij}^0}\right],$$

$$\text{s.t} \quad \inf_{\mathcal{F}_k\in\mathcal{D}_k}\mathbb{P}_{\mathcal{F}_k}\left(\sum_{i=1}^{I_k}c_i^k\prod_{j=1}^{M}t_j^{a_{ij}^k}\leq 1\right)\geq y_k,\,,k=1,....,K,$$

$$\prod_{k=1}^{K}y_k\geq 1-\epsilon,\ 0<y_k\leq 1\,,\ k=1,...,K.$$

By Theorem 1 in [89], we conclude that (JCP) is equivalent to (JCP_{ind}). □

Problem (JCP_{ind}) is not convex. By applying the logarithmic transformation $r_j = log(t_j)$, $j=1,...,M$ and $x_k = \log(y_k)$, $k=1,...,K$, we have the following equivalent reformulation of (JCP_{ind})

$$(\text{JCP}_{ind}^{log}) \min_{r\in\mathbb{R}^M, x\in\mathbb{R}^K} \sum_{i=1}^{I_0}\mu_i^0\exp\left\{\sum_{j=1}^{M}a_{ij}^0r_j\right\}+\sqrt{\gamma_1^0}\sqrt{\sum_{i=1}^{I_0}\sum_{l=1}^{I_0}\sigma_{i,l}^0\exp\left\{\sum_{j=1}^{M}(a_{ij}^0+a_{lj}^0)r_j\right\}}, \tag{6.5}$$

$$\text{s.t} \quad \sum_{i=1}^{I_k}\mu_i^k\exp\left\{\sum_{j=1}^{M}a_{ij}^kr_j\right\}+\sqrt{\gamma_1^k}\sqrt{\sum_{i=1}^{I_k}\sum_{l=1}^{I_k}\sigma_{i,l}^k\exp\left\{\sum_{j=1}^{M}(a_{ij}^k+a_{lj}^k)r_j\right\}}$$

$$+\sqrt{\gamma_2^k}\sqrt{\sum_{i=1}^{I_k}\sum_{l=1}^{I_k}\sigma_{i,l}^k\exp\left\{\sum_{j=1}^{M}(a_{ij}^k+a_{lj}^k)r_j+\log\left(\frac{e^{x_k}}{1-e^{x_k}}\right)\right\}}\leq 1,\ k=1,...,K,$$

$$\sum_{k=1}^{K}x_k\geq\log(1-\epsilon),\ x_k\leq 0\,,\ k=1,...,K. \tag{6.6}$$

Theorem 50. [89] If $\sigma_{i,l}^k\geq 0$ for all i, l and k, problem (JCP_{ind}^{log}) is a convex programming problem.

Case (JCP) with Jointly Dependent Row Vectors.

In this case, (JCP) is equivalent to [89]

$$(\text{JCP}_{dep}) \min_{t\in\mathbb{R}^M_{++}, y\in\mathbb{R}^K_+} \sum_{i=1}^{I_0}\mu_i^0\prod_{j=1}^{M}t_j^{a_{ij}^0}+\sqrt{\gamma_1^0}\sqrt{\sum_{i=1}^{I_0}\sum_{l=1}^{I_0}\sigma_{i,l}^0\prod_{j=1}^{M}t_j^{a_{ij}^0+a_{lj}^0}}, \tag{6.7}$$

$$\text{s.t} \quad \sum_{i=1}^{I_k}\mu_i^k\prod_{j=1}^{M}t_j^{a_{ij}^k}+\sqrt{\gamma_1^k}\sqrt{\sum_{i=1}^{I_k}\sum_{l=1}^{I_k}\sigma_{i,l}^k\prod_{j=1}^{M}t_j^{a_{ij}^k+a_{lj}^k}}$$

$$+\sqrt{\frac{y_k}{1-y_k}}\sqrt{\gamma_2^k}\sqrt{\sum_{i=1}^{I_k}\sum_{l=1}^{I_k}\sigma_{i,l}^k\prod_{j=1}^{M}t_j^{a_{ij}^k+a_{lj}^k}}\leq 1,\ k=1,...,K, \tag{6.8}$$

$$\sum_{k=1}^{K}y_k\geq K-\epsilon,\ 0<y_k\leq 1\ , k=1,...,K. \tag{6.9}$$

As for the independent case, we obtain the following biconvex equivalent problem for (JCP_{dep})

$$(\text{JCP}_{dep}^{log}) \min_{r\in\mathbb{R}^M, x\in\mathbb{R}^K} \sum_{i=1}^{I_0}\mu_i^0\exp\left\{\sum_{j=1}^{M}a_{ij}^0r_j\right\}+\sqrt{\gamma_1^0}\sqrt{\sum_{i=1}^{I_0}\sum_{l=1}^{I_0}\sigma_{i,l}^0\exp\left\{\sum_{j=1}^{M}(a_{ij}^0+a_{lj}^0)r_j\right\}}, \tag{6.10}$$

$$\text{s.t} \quad \sum_{i=1}^{I_k}\mu_i^k\exp\left\{\sum_{j=1}^{M}a_{ij}^kr_j\right\}+\sqrt{\gamma_1^k}\sqrt{\sum_{i=1}^{I_k}\sum_{l=1}^{I_k}\sigma_{i,l}^k\exp\left\{\sum_{j=1}^{M}(a_{ij}^k+a_{lj}^k)r_j\right\}}$$

$$+\sqrt{\gamma_2^k}\sqrt{\sum_{i=1}^{I_k}\sum_{l=1}^{I_k}\sigma_{i,l}^k\exp\left\{\sum_{j=1}^{M}(a_{ij}^k+a_{lj}^k)r_j+\log\left(\frac{y_k}{1-y_k}\right)\right\}}\leq 1,\ k=1,...,K,$$

$$\sum_{k=1}^{K}y_k\geq K-\epsilon,\ 0<y_k\leq 1\ , k=1,...,K. \tag{6.11}$$

Theorem 51. [89] If $\epsilon\leq 0.5$ and $\sigma_{i,l}^k\geq 0$ for all i, l and k, problem (JCP_{dep}^{log}) is a convex programming problem.

6.2.2 . Uncertainty Sets with Known First Order Moment and Nonnegative Support

In this section, we consider uncertainty sets with nonnegative supports and known first-order moments. The uncertainty sets for (JCP) can be formulated as follows

$$\mathcal{D}_k^3(\mu_k,\Sigma_k)=\left\{\mathcal{F}_k\ \middle|\ \begin{array}{c}\mathbb{E}[C^k]=\mu^k\\ \mathbb{P}_{\mathcal{F}_k}[C^k\geq 0]=1\end{array}\right\}, k=0,1,...,K, \text{ where } \mu^k>0.$$

Case (JCP) with Jointly Independent Row Vectors.

We first consider the case when the marginal distributions in the uncertainty set are jointly independent. Using the strong duality [89], (JCP) can be reformulated as follows

$$(\text{JCP}^{\text{ind}}_{NS}) \quad \min_{t\in\mathbb{R}^M_{++},\lambda,\beta,\pi} \sum_{i=1}^{I_0} \mu_i^0 \prod_{j=1}^{M} t_j^{a_{ij}^0}, \tag{6.12}$$

$$\text{s.t} \quad \prod_{k=1}^{K} y_k \geq 1-\epsilon,\ 0 \leq y_k \leq 1\ , k=1,...,K, \tag{6.13}$$

$$y_k\lambda_k^{-1} - \lambda_k^{-1}\beta^{kT}\mu^k \leq 1, k=1,....,K, \tag{6.14}$$

$$\beta_k \leq 0, 0 < \lambda \leq 1, k=1,....,K, \tag{6.15}$$

$$\lambda_k^{-1}\pi_k \geq 1, k=1,....,K, \tag{6.16}$$

$$(-\beta_k)^{-1}\pi_k \prod_{j=1}^{M} t_j^{a_{ij}^k} \leq 1, i=1,...,I_k, k=1,....,K, \tag{6.17}$$

(JCP) can be reformulated as a convex problem using a logarithmic transformation $x_j = \log(y_j)$, $t_j = \log(r_j)$, $\tilde{\lambda_k} = \log(\lambda_k)$, $\tilde{\beta}_k = \log(-\beta_k)$, $\tilde{\pi} = \log(\pi)$. Problem (JCP_{NS}) becomes,

$$(\text{JCP}^{log}_{NS-ind}) \quad \min_{x,r,\tilde{\lambda},\tilde{\beta},\tilde{\pi}} \sum_{i=1}^{I_0} \mu_i^0 \exp\left\{\sum_{j=1}^{M} a_{ij}^0 r_j\right\}, \tag{6.18}$$

$$\text{s.t} \quad \sum_{k=1}^{K} x_k \geq \log(1-\epsilon),\ x_k \leq 0, k=1,...,K, \tag{6.19}$$

$$\exp(x_k - \tilde{\lambda}_k) + \sum_{i=1}^{I_k} \exp\left\{-\tilde{\lambda}_k + \tilde{\beta}_i^k + \log\mu_i^k\right\} \leq 1, k=1,....,K,$$

$$\tilde{\lambda}_k \leq 0, k=1,....,K, \tag{6.20}$$

$$\tilde{\lambda}_k \leq \tilde{\pi}_k, k=1,....,K, \tag{6.21}$$

$$\tilde{\pi}_k + \sum_{j=1}^{M} a_{ij}^k r_j - \tilde{\beta}_i^k \leq 0, i=1,...,I_k, k=1,....,K. \tag{6.22}$$

Case (JCP) with Jointly Dependent Row Vectors.

In the case where the constraints of (JCP) are jointly dependent, we have the following deterministic equivalent

$$(\text{JCP}_{NS}^{\text{dep}}) \min_{t\in\mathbb{R}^M_{++},\lambda,\beta,\pi} \sum_{i=1}^{I_0} \mu_i^0 \prod_{j=1}^{M} t_j^{a_{ij}^0}, \tag{6.23}$$

$$\text{s.t} \quad \prod_{k=1}^{K} y_k \geq K-\epsilon,\ 0\leq y_k \leq 1\ , k=1,...,K, \tag{6.24}$$

$$y_k\lambda_k^{-1} - \lambda_k^{-1}\beta^{kT}\mu^k \leq 1, k=1,....,K, \tag{6.25}$$

$$\beta_k \leq 0, 0<\lambda\leq 1, k=1,....,K, \tag{6.26}$$

$$\lambda_k^{-1}\pi_k \geq 1, k=1,....,K, \tag{6.27}$$

$$(-\beta_k)^{-1}\pi_k \prod_{j=1}^{M} t_j^{a_{ij}^k} \leq 1, i=1,...,I_k, k=1,....,K, \tag{6.28}$$

The log transformation fails to convert (JCP_{NS}^{log}) into a convex problem. Nevertheless, we can obtain the following biconvex equivalent by taking $t_j = \log(r_j)$, $\tilde{\lambda_k} = \log(\lambda_k)$, $\tilde{\beta_k} = \log(-\beta_k)$, $\tilde{\pi} = \log(\pi)$

$$(\text{JCP}_{NS-dep}^{log}) \min_{x,r,\tilde{\lambda},\tilde{\beta},\tilde{\pi}} \sum_{i=1}^{I_0} \mu_i^0 \exp\left\{\sum_{j=1}^{M} a_{ij}^0 r_j\right\}, \tag{6.29}$$

$$\text{s.t} \quad \prod_{k=1}^{K} y_k \geq K-\epsilon,\ 0\leq y_k \leq 1\ , k=1,...,K, \tag{6.30}$$

$$y_k \exp(-\tilde{\lambda}_k) + \sum_{i=1}^{I_k} \exp\left\{-\tilde{\lambda}_k + \tilde{\beta}_i^k + \log\mu_i^k\right\} \leq 1, k=1,....,K,$$

$$\tilde{\lambda}_k \leq 0, k=1,....,K, \tag{6.31}$$

$$\tilde{\lambda}_k \leq \tilde{\pi}_k, k=1,....,K, \tag{6.32}$$

$$\tilde{\pi}_k + \sum_{j=1}^{M} a_{ij}^k r_j - \tilde{\beta}_i^k \leq 0, i=1,...,I_k, k=1,....,K. \tag{6.33}$$

6.2.3 . A dynamical recurrent neural network for (JCP_{dep}^{log}), (JCP_{ind}^{log}) and $(\text{JCP}_{NS-ind}^{log})$

Observe that (JCP_{dep}^{log}), (JCP_{ind}^{log}) and $(\text{JCP}_{NS_ind}^{log})$ can be written in the following general form

$$\min_r f(z), \tag{6.34}$$
$$\text{s.t. } g(z)\leq 0,$$

where f and g are two convex functions.

For (JCP$^{log}_{ind}$), $z = (r, x)^T$, $f(z) = \sum_{i=1}^{I_0} \mu_i^0 \exp\left\{\sum_{j=1}^{M} a_{ij}^0 r_j\right\} + \sqrt{\gamma_1^0}\sqrt{\sum_{i=1}^{I_0}\sum_{l=1}^{I_0} \sigma_{i,l}^0 \exp\left\{\sum_{j=1}^{M}(a_{ij}^0 + a_{lj}^0) r_j\right\}}$

and $g(z) = \begin{pmatrix} \sum_{i=1}^{I_1} \mu_i^1 \exp\left\{\sum_{j=1}^{M} a_{ij}^1 r_j\right\} + \sqrt{\gamma_1^1}\sqrt{\sum_{i=1}^{I_1}\sum_{l=1}^{I_1} \sigma_{i,l}^1 \exp\left\{\sum_{j=1}^{M}(a_{ij}^1 + a_{lj}^1) r_j\right\}} \\ + \sqrt{\frac{e^{x_1}}{1-e^{x_1}}}\sqrt{\gamma_2^1}\sqrt{\sum_{i=1}^{I_1}\sum_{l=1}^{I_1} \sigma_{i,l}^1 \exp\left\{\sum_{j=1}^{M}(a_{ij}^1 + a_{lj}^1) r_j\right\}} - 1 \\ \vdots \\ \sum_{i=1}^{I_K} \mu_i^K \exp\left\{\sum_{j=1}^{M} a_{ij}^K r_j\right\} + \sqrt{\gamma_1^K}\sqrt{\sum_{i=1}^{I_K}\sum_{l=1}^{I_K} \sigma_{i,l}^K \exp\left\{\sum_{j=1}^{M}(a_{ij}^K + a_{lj}^K) r_j\right\}} + \\ \sqrt{\frac{e^{x_K}}{1-e^{x_K}}}\sqrt{\gamma_2^K}\sqrt{\sum_{i=1}^{I_K}\sum_{l=1}^{I_K} \sigma_{i,l}^K \exp\left\{\sum_{j=1}^{M}(a_{ij}^K + a_{lj}^K) r_j\right\}} - 1 \\ \log(1-\epsilon) - \sum_{k=1}^{K} x_k \\ x_1 \\ \vdots \\ x_K \end{pmatrix}$.

For (JCP$^{log}_{dep}$), $z = (r, x)^T$, $f(z) = \sum_{i=1}^{I_0} \mu_i^0 \exp\left\{\sum_{j=1}^{M} a_{ij}^0 r_j\right\} + \sqrt{\gamma_1^0}\sqrt{\sum_{i=1}^{I_0}\sum_{l=1}^{I_0} \sigma_{i,l}^0 \exp\left\{\sum_{j=1}^{M}(a_{ij}^0 + a_{lj}^0) r_j\right\}}$

and $g(z) = \begin{pmatrix} \sum_{i=1}^{I_1} \mu_i^1 \exp\left\{\sum_{j=1}^{M} a_{ij}^1 r_j\right\} + \sqrt{\gamma_1^1}\sqrt{\sum_{i=1}^{I_1}\sum_{l=1}^{I_1} \sigma_{i,l}^1 \exp\left\{\sum_{j=1}^{M}(a_{ij}^1 + a_{lj}^1) r_j\right\}} \\ + \sqrt{\frac{e^{x_1}}{1-e^{x_1}}}\sqrt{\gamma_2^1}\sqrt{\sum_{i=1}^{I_1}\sum_{l=1}^{I_1} \sigma_{i,l}^1 \exp\left\{\sum_{j=1}^{M}(a_{ij}^1 + a_{lj}^1) r_j\right\}} - 1 \\ \vdots \\ \sum_{i=1}^{I_K} \mu_i^K \exp\left\{\sum_{j=1}^{M} a_{ij}^K r_j\right\} + \sqrt{\gamma_1^K}\sqrt{\sum_{i=1}^{I_K}\sum_{l=1}^{I_K} \sigma_{i,l}^K \exp\left\{\sum_{j=1}^{M}(a_{ij}^K + a_{lj}^K) r_j\right\}} + \\ \sqrt{\frac{e^{x_K}}{1-e^{x_K}}}\sqrt{\gamma_2^K}\sqrt{\sum_{i=1}^{I_K}\sum_{l=1}^{I_K} \sigma_{i,l}^K \exp\left\{\sum_{j=1}^{M}(a_{ij}^K + a_{lj}^K) r_j\right\}} - 1 \\ \log(K-\epsilon) - \sum_{k=1}^{K} x_k \\ x_1 \\ \vdots \\ x_K \end{pmatrix}$.

For (JCP$^{log}_{NS-ind}$), $z = (r, x, \tilde{\lambda}, \tilde{\beta}, \tilde{\pi})^T$, $f(z) = \sum_{i=1}^{I_0} \mu_i^0 \prod_{j=1}^{M} t_j^{a_{ij}^0}$

and $g(z) = \begin{pmatrix} \log(1-\epsilon) - \sum_{k=1}^{K} x_k \\ x_1 \\ \vdots \\ x_K \\ \exp(x_1 - \tilde{\lambda}_1) + \sum_{i=1}^{I_1} \exp\left\{-\tilde{\lambda}_1 + \tilde{\beta}_i^1 + \log\mu_i^1\right\} - 1 \\ \vdots \\ \exp(x_K - \tilde{\lambda}_K) + \sum_{i=1}^{I_K} \exp\left\{-\tilde{\lambda}_K + \tilde{\beta}_i^K + \log\mu_i^K\right\} - 1 \\ \tilde{\lambda}_1 \\ \vdots \\ \tilde{\lambda}_1 - \tilde{\pi}_1 \\ \vdots \\ \tilde{\lambda}_K - \tilde{\pi}_K \\ \tilde{\pi}_1 + \sum_{j=1}^{M} a_{ij}^1 r_j - \tilde{\beta}_i^1 \leq 0, i = 1, ..., I_1 \\ \vdots \\ \tilde{\pi}_K + \sum_{j=1}^{M} a_{ij}^K r_j - \tilde{\beta}_i^K \leq 0, i = 1, ..., I_K \end{pmatrix}$.

We know that z^* is an optimal solution of (6.34) if and only if the following Karush–Kuhn–Tucker (KKT) are satisfied.

$$\nabla f(z) + \nabla g(z)^T \gamma = 0 \tag{6.35}$$

$$\gamma \geq 0, \gamma^T g(z) = 0 \tag{6.36}$$

To solve problem (6.34), we propose a dynamical recurrent neural network driven by the following ODE system

$$\kappa \frac{dz}{dt} = -\left(\nabla f(z) + \nabla g(z)^T \left(\gamma + g(z)\right)_+\right) \tag{6.37}$$

$$\kappa \frac{d\gamma}{dt} = -\gamma + \left(\gamma + g(z)\right)_+ \tag{6.38}$$

where $z(.)$ and $\gamma(.)$ are two time-dependent variable, κ is a given convergence rate and $(x)_+ = \max(x, 0)$.

Theorem 52. If (z^*, γ^*) is an equilibrium point of (6.37)-(6.38) if and only if z^* is an optimal solution of (6.34) where γ^* is the correspondent Lagrange multiplier.

Proof. Let (z^*, γ^*) is an equilibrium point of (6.37)-(6.38), then $\frac{dz^*}{dt} = 0$ and $\frac{d\gamma^*}{dt} = 0$.

$$\frac{dz^*}{dt} = 0 \Leftrightarrow \nabla f(z^*) + \nabla g(z^*)^T \left(\gamma^* + g(z^*)\right)_+ = 0, \tag{6.39}$$

$$\frac{d\gamma^*}{dt} = 0 \Leftrightarrow -\gamma^* + \left(\gamma^* + g(z^*)\right)_+ = 0 \tag{6.40}$$

Observe that $\gamma^* = (\gamma^* + g(z^*))_+$ if and only if $\gamma^* \geq 0$, $g(z^*) \leq 0$ and $\gamma^{*T} g(z^*) = 0$, we obtain then (6.130) of the KKT system (6.130)- (6.36). Furthermore, we replace $(\gamma^* + g(z^*))_+$ by γ^* in the right hand side of (6.39) we obtain then $\nabla f(z^*) + \nabla g(z^*)^T \gamma^* = 0$ which is equation (6.36) of the KKT system (6.130)- (6.36). For the converse part of the theorem, it is straightforward that if z^* is an optimal solution of (6.34) where γ^* is the correspondent Lagrange multiplier then (z^*, γ^*) is an equilibrium point of (6.37)-(6.38). □

Lemma 53. For any initial point $(z(t_0), \gamma(t_0))$, there exists a unique continuous solution $(z(t), \gamma(t))$ for (6.37)-(6.38).

Proof. The right-hand side of system (6.37)-(6.38) is locally Lipschitz continuous, given that ∇f, ∇g and $(\gamma + g)_+$ and are locally Lipschitz continuous. By applying the local existence theorem of ordinary differential equations, we can conclude that there exists a unique continuous solution trajectory $(z(t), \gamma(t))$ for (6.37)-(6.38). □

Theorem 54. The neural network proposed in equations (6.37)-(6.38) exhibits global stability in the Lyapunov sense. Furthermore, the dynamical network globally converges to a KKT point denoted (z^*, γ^*) where z^* is the optimal solution of the problem (6.34).

Proof. Let $\zeta = (z, \gamma)$, we define $U(\zeta) = \begin{bmatrix} -\left(\nabla f(z) + \nabla g(z)^T (\gamma + g(z))_+\right) \\ -\gamma + (\gamma + g(z))_+ \end{bmatrix}$.
First, consider the following Lyapunov function

$$E(\zeta) = ||U(\zeta)||^2 + \frac{1}{2}||\zeta - \zeta^*||, \tag{6.41}$$

where $\zeta^* = (z^*, \gamma^*)$ is an equilibrium point of (6.37)-(6.38).
$\frac{dE(\zeta(t))}{dt} = \frac{dU}{dt}^T U + U^T \frac{dU}{dt} + (\zeta - \zeta^*)^T \frac{d\zeta}{dt}$. Observe that $\frac{dU}{dt} = \frac{dU}{d\zeta} \times \frac{d\zeta}{dt} = \nabla U(\zeta) U(\zeta)$.
Without loss of generality suppose that there exists $p \in \mathbb{N}$ such that $(\gamma + g(z))_+ = (\gamma_1 + g_1(z)), ..., (\gamma_p + g_p(z)), 0, ..., 0)$, and we define $g^p = (g_1, ..., g_p)$.
We have $\nabla U(\zeta) = \begin{bmatrix} -\left(\nabla^2 f(z) + \sum_{i=1}^p \nabla^2 g^p(z) (\gamma_p + g_p(z)) + \nabla g(z)^T \nabla g(z)\right) & -\nabla g^p(z)^T \\ \nabla g^p(z) & S_p \end{bmatrix}$.
where $S_p = \begin{bmatrix} O_{p\times p} & O_{p\times(N-p)} \\ O_{(N-p)\times q} & I_{(N-p)\times(N-p)} \end{bmatrix}$, where N is the length of vector γ.
Since f and g are convex, then the hessian matrices $\nabla^2 f$ and $\nabla^2 g^p$ are positive semidefinite. Furthermore $\nabla g^T \nabla g$ is positive semidefinite, we conclude that ∇U is negative semidefinite.
Back to the expression of $\frac{dE(\zeta(t))}{dt}$,
we have $\frac{dE(\zeta(t))}{dt} = \underbrace{U^T(\nabla U + \nabla U^T)U}_{\leq 0 \text{ since } \nabla U \text{ is negative semidfinite}} + \underbrace{(\zeta - \zeta^*)^T (U(\zeta) - U(\zeta^*))}_{\leq 0 \text{ by Lemma 4 in [132]}} \leq 0$. So the neural network (6.37)-(6.38) is globally stable in the sense of Lyapunov.
Next similarly to the proof of Theorem 5 in [132], we prove that the dynamical neural network (6.37)-(6.38) is globally convergent to (z^*, γ^*) where z^* is the optimal solution of (6.34).

□

6.2.4 . A two-time scale neurodynamic duplex for (JCP$^{log}_{NS-dep}$)

(JCP$^{log}_{NS-dep}$) can be written in the following general form

$$\begin{aligned} &\min_{z,y} f(z), \\ &\text{s.t. } g(z,y) \leq 0, \end{aligned} \tag{6.42}$$

where f is a convex function and g is a biconvex function, $z = (r, \tilde{\lambda}, \tilde{\beta}, \tilde{\pi})^T$, $f(z) = \sum\limits_{i=1}^{I_0} \mu_i^0 \prod_{j=1}^{M} t_j^{a_{ij}^0}$

and $g(z,y) = \begin{pmatrix} K - \epsilon - \prod\limits_{k=1}^{K} y_k \\ -y_1 \\ \vdots \\ -y_K \\ y_1 - 1 \\ \vdots \\ y_K - 1 \\ y_1 \exp(-\tilde{\lambda}_1) + \sum\limits_{i=1}^{I_1} \exp\left\{-\tilde{\lambda}_1 + \tilde{\beta}_i^1 + \log\mu_i^1\right\} - 1 \\ \vdots \\ y_K \exp(-\tilde{\lambda}_K) + \sum\limits_{i=1}^{I_K} \exp\left\{-\tilde{\lambda}_K + \tilde{\beta}_i^K + \log\mu_i^K\right\} - 1 \\ \tilde{\lambda}_1 \\ \vdots \\ \tilde{\lambda}_1 - \tilde{\pi}_1 \\ \vdots \\ \tilde{\lambda}_K - \tilde{\pi}_K \\ \tilde{\pi}_1 + \sum_{j=1}^{M} a_{ij}^1 r_j - \tilde{\beta}_i^1 \leq 0, i = 1, ..., I_1 \\ \vdots \\ \tilde{\pi}_K + \sum_{j=1}^{M} a_{ij}^K r_j - \tilde{\beta}_i^K \leq 0, i = 1, ..., I_K \end{pmatrix}$.

We denote $\mathcal{U} = \{z, y \mid g(z,y) \leq 0\}$ the feasible set of (6.42).The Lagrangian function of problem (6.42) is defined as follows:

$$\mathcal{L}(z, y, \omega) = f(z) + \omega^T g(z, y). \tag{6.43}$$

For any $(z,y) \in \mathcal{U}$, the KKT conditions are stated as follows:

$$\nabla L(z, y, \omega) = 0, \tag{6.44}$$

$$\omega \geq 0, \omega^T g(z, y) = 0. \tag{6.45}$$

Definition 21. Let $(z,y) \in \mathcal{U}$, (z,y) is called a partial optimum of (6.42) if and only if

$$f(z) \leq f(\tilde{z}), \forall \tilde{z} \in \mathcal{U}_y, \tag{6.46}$$

where $\mathcal{U}_y = \{z \mid g(z,y) \leq 0\}$.

Theorem 55. The KKT system (6.44)-(6.45) is equivalent to the following system

$$\nabla f(z) + \nabla_z g(z,y)^T(\omega + g(x,z))_+ = 0 \tag{6.47}$$
$$\nabla_y g(z,y)^T(\omega + g(z,y))_+ = 0 \tag{6.48}$$
$$(\omega + g(x,z))_+ - \omega = 0 \tag{6.49}$$

Proof. The proof of Theorem 55 follows the same lines of the proof of Theorem 52. □

Based on the equations (6.47)-(6.49), we consider the following two-time-scale recurrent neural network model

$$\kappa_1 \frac{dz}{dt} = -\left(\nabla f(z) + \nabla_z g(z,y)^T(\omega + g(x,z))_+\right), \tag{6.50}$$
$$\kappa_2 \frac{dy}{dt} = -\left(\nabla_y g(z,y)^T(\omega + g(z,y))_+\right), \tag{6.51}$$
$$\kappa_2 \frac{d\omega}{dt} = -\omega + (\lambda + g(z,y))_+, \tag{6.52}$$

where (z,y,ω) are now time-dependent variables and κ_1 and κ_2 are two time scaling constants with $\kappa_1 \neq \kappa_2$. We propose a duplex of two two-time-scale recurrent neural network (6.50)-(6.52) for solving (6.34) one with $\kappa_1 > \kappa_2$ and the second with $\kappa_1 < \kappa_2$ as shown in Figure 6.6.

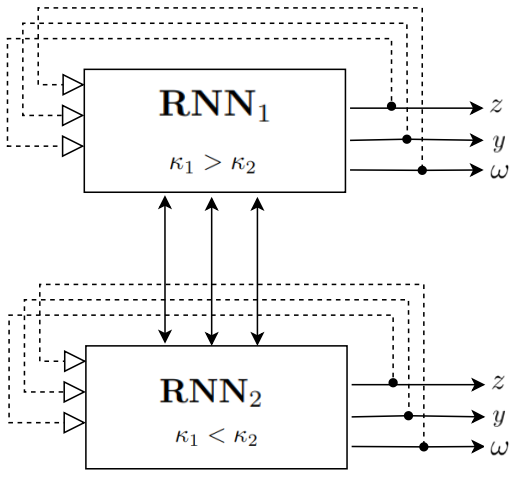

Figure 6.1: A block diagram depicting a duplex neurodynamic system with a two-timescale configuration

Theorem 56. (z,y,ω) is an equilibrium point of (6.50)-(6.52) if and only if (z,y,ω) is a KKT point of (6.42).

Proof. Let (z, y, ω) is an equilibrium point of (6.50)-(6.52). We have then

$$\frac{dz}{dt} = 0 \Leftrightarrow -\left(\nabla f(z) + \nabla_z g(z,y)^T(\omega + g(x,z))_+\right) = 0, \tag{6.53}$$

$$\frac{dy}{dt} = 0 \Leftrightarrow -\left(\nabla_y g(z,y)^T(\omega + g(z,y))_+\right) = 0, \tag{6.54}$$

$$\frac{d\omega}{dt} = 0 \Leftrightarrow -\omega + (\lambda + g(z,y))_+ = 0. \tag{6.55}$$

We obtain then system (6.47)-(6.49). By Theorem 55, the conclusion follows. The converse part of the Theorem is traightforward. □

The process begins by initializing the state variables of the neurodynamic models. Subsequently, each model undergoes a precise local search based on its dynamics to optimize its performance. Once all neurodynamic models have converged to their equilibria, the initial states of the recurrent neural networks are optimized using the particle swarm optimization (PSO) updating rule. In this context, we represent the position of the i^{th} particle as $\Lambda_i = (\Lambda_{i1}, ..., \Lambda_{in})^T$, and its velocity as $v_i = (v_{i1}, ..., v_{in})^T$. The inertia weight $w \in [0, 1]$ determines the extent to which the particle retains its previous velocity. The best previous position that yielded the maximum fitness value for the i^{th} particle is denoted as $\tilde{\Lambda}_i = (\tilde{\Lambda}_{i1}, ..., \tilde{\Lambda}_{in})^T$, and the best position in the entire swarm that yielded the maximum fitness value is represented by $\hat{\Lambda} = (\hat{\Lambda}_1, ..., \hat{\Lambda}_n)^T$. The initial state of each neurodynamic model is updated using the PSO updating rule, as described in reference [34].

$$v_i(j+1) = wv_i(j) + c_1 r_1(\tilde{\Lambda}_i - \Lambda_i(j)) + c_2 r_2(\hat{\Lambda}_i - \Lambda_i(j)), \tag{6.56}$$

$$\Lambda_i(j+1) = \Lambda_i(j) + v_i(j+1). \tag{6.57}$$

where the iterative index is represented by j, while the two weighting parameters are denoted as c_1 and c_2 and r_1 and r_2 represent two random values from the interval $[0, 1]$.

To achieve global optimization, the diversity of initial neuronal states is crucial. One approach to enhance this diversity is by introducing a mutation operator, which generates a random $\Lambda_i(j+1)$. This random generation of $\Lambda_i(j+1)$ helps increase the variation among the initial neuronal states. To measure the diversity of these states, we employ the following function

$$d = \frac{1}{n}\sum_{i=1}^{n} \|\Lambda_i(j+1) - \hat{\Lambda}(j)\|. \tag{6.58}$$

We utilize the wavelet mutation operator proposed in [86], which is performed for the i-th particle if $d < \zeta$. The mutation operation is carried out as follows

$$\Lambda_i(j+1) = \begin{cases} \Lambda_i(j) + \mu(h_i - \Lambda_i(j)) & , \mu > 0 \\ \Lambda_i(j) + \mu(\Lambda_i(j) - l_i) & , \mu < 0 \end{cases} \tag{6.59}$$

where h_i and l_i are the upper and the lower bound for Λ_i, respectively. $\zeta > 0$ is a given threshold and μ is defined using a walvet function

$$\mu = \frac{1}{\sqrt{a}} e^{-\frac{\phi}{2a}} \cos(5\frac{\phi}{a}) \tag{6.60}$$

When the value of μ approaches 1, the mutated element of the particle tends to move towards the maximum value of $\Lambda_i(j+1)$. On the other hand, as μ approaches -1, the mutated element tends to move towards the minimum value of $x_i(j+1)$. The magnitude of $|\mu|$ determines the size of the search space for $x_i(j+1)$, with larger values indicating a wider search space. Conversely, smaller values of $|\mu|$ result in a smaller search space, allowing for fine-tuning.

To achieve fine-tuning, the dilation parameter a is adjusted based on the current iteration j relative to the total number of iterations T. Specifically, a is set as a function of j/T, with $a = e^{10\frac{j}{T}}$. Additionally, ϕ is randomly generated from the interval $[-2.5a, 2.5a]$.

The algorithm details are given in Algorithm 2 where $\Lambda = (z, y, \omega)$

Algorithm 2 The neurodynamic duplex

- Let $\Lambda_1(0)$ and $\Lambda_2(0)$ be randomly generated in the feasible region.
- Let $\tilde{\Lambda}(0) = \hat{\Lambda}(0) = y(0)$ the initial best previous position and best position, respectively.
-Set the convergence error ϵ.

while $||\Lambda(j+1) - \Lambda(j)|| \geq \epsilon$ **do**
  Compute the equilibrium points $\bar{\Lambda}_1(j)$ and $\bar{\Lambda}_2(j)$ of RNN$_1$ and RNN$_2$.
  if $f(\bar{z}_1(j)) < f(\tilde{z}(j))$ **then**
    $\tilde{\Lambda}(j+1) = \bar{\Lambda}_1(j)$
  else
    $\tilde{\Lambda}(j+1) = \tilde{\Lambda}(j)$
  end if
  if $f(\bar{z}_2(j)) < f(\tilde{z}(j))$ **then**
    $\tilde{\Lambda}(j+1) = \bar{\Lambda}_2(j)$
  else
    $\tilde{\Lambda}(j+1) = \tilde{\Lambda}(j)$
  end if
  if $f(\tilde{z}(j)) < f(\hat{z}(j))$ **then**
    $\hat{\Lambda}(j+1) = \tilde{\Lambda}(j+1)$
  else
    $\hat{\Lambda}(j+1) = \hat{\Lambda}(j)$
  end if
  Compute the value of $\Lambda(j+1)$ following (6.56)-(6.57).
  if $d < \zeta$ **then**
    Perform the wavelet mutation (6.59).
  end if
  j=j+1
end while

Lemma 57. [137] Suppose that the objective function f is measurable, and the feasible region $\mathcal{U}$ is a measurable subset, and for any Borel subset $\mathcal{B}$ of $\mathcal{U}$ with positive Lebesgue

measure we have $\prod_{k=1}^{\infty}(1-\mathbb{P}_k(\mathcal{B}))=0$. Let $\{y(k)\}_{k=1}^{\infty}$ be a sequence generated by a stochastic optimization algorithm. If $\{y(k)\}_{k=1}^{\infty}$ is a nonincreasing sequence, then converges with probability one to the global optimum set.

Theorem 58. If the state of the neurodynamic model with a single timescale, described by the following equations

$$\kappa\frac{dz}{dt}=-\left(\nabla f(z)+\nabla_z g(z,y)^T(\omega+g(x,z))_+\right), \tag{6.61}$$

$$\kappa\frac{dy}{dt}=-\left(\nabla_y g(z,y)^T(\omega+g(z,y))_+\right), \tag{6.62}$$

$$\kappa\frac{d\omega}{dt}=-\omega+(\lambda+g(z,y))_+, \tag{6.63}$$

converges to an equilibrium point, then the state of the neurodynamic model with two timescales, as described by equations (6.50)-(6.52), globally converges to a partial optimum of problem (6.42).

Proof. We recall the Lagrangian function of (6.42)

$$\mathcal{L}(z,y,\omega)=f(z)+\omega^T g(z,y). \tag{6.64}$$

An equilibrium point (z^*,y^*,ω^*) of (6.61)-(6.63) corresponds to a KKT point of (6.42). We fix y^*, and take $z\in\mathcal{U}_{y^*}$, (6.42) becomes a convex optimization problem and we have

$$\mathcal{L}(z^*,y^*,\omega^*)\leq\mathcal{L}(z,y^*,\omega^*), \tag{6.65}$$

which is equivalent to

$$f(z^*)+{\omega^*}^T g(z^*,y^*)\leq f(z)+{\omega^*}^T g(z,y^*). \tag{6.66}$$

As ${\omega^*}^T g(z,y^*)\leq{\omega^*}^T g(z^*,y^*)=0$, we have $f(z^*)\leq f(z)$. By Definition 21, (z^*,y^*) is a partial optimum of 6.42. □

Theorem 59. The duplex of two two-timescale neural networks in Figure 6.6 system is globally convergent to a global optimal solution of problem (6.34) with probability one.

Proof. By Theorem 58, the two-timescale neurodynamic models RNN_1 and RNN_2 are proven to converge to a partial optimum. From Algorithm 2, the solution sequence is generated as follows

$$\begin{cases}\hat{\Lambda}(j+1) & =\tilde{\Lambda}(j+1)\text{ if }f(\tilde{z}(j))<f(\hat{z}(j)),\\ \hat{\Lambda}(j+1) & =\hat{\Lambda}(j)\text{ else.}\end{cases}$$

We observe that the generated solution sequence is monotonically increasing $\{f(\tilde{\Lambda}(j))\}_{j=1}^{\infty}$. Let $\mathcal{M}i,j$ represent the supporting set of the initial state of RNNi at iteration j. According to equation (6.59), the mutation operation ensures that the initial states of the recurrent

neural networks are constrained to the feasible region $\mathcal{U}$. Therefore, for every iteration index $J \geq 1$, the supporting sets satisfy the following condition:

$$\mathcal{U} \subseteq \mathcal{M} = \bigcup_{j=1}^{J} \bigcup_{i=1}^{2} \mathcal{M}_{i,j}. \tag{6.67}$$

Consequently, we have $v(\mathcal{U}) = v(\mathcal{M}) > 0$.
By Lemma 66, we have

$$\lim_{j->\infty} \mathbb{P}(\hat{\Lambda}(j) \in \Phi) = 1 \tag{6.68}$$

where Φ is the set of the global optimal solutions of (6.34). The conclusion follows.

□

6.2.5 . Numerical experiments

We consider three geometric problems to evaluate the performance of our neurodynamic approaches. All the algorithms in this Section are implemented in Python. We run our algorithms on Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz. The random instances are generated with numpy.random, and we solve the ODE systems with solve_ivp of scipy.integrate. The deterministic equivalent programs are solved with the package gekko and the gradients and partial derivatives are computed with autograd.grad and autograd.jacobian. For the following numerical experiments, we set $\gamma_1^k = 2$, $\gamma_2^k = 2$ and the error tolerance for the neurodynamic duplex $\zeta = 10^{-4}$. In the second subsection, we evaluate the quality of our neurodynamic duplex by comparing the obtained solutions with the ones given by the Convex Alternate Search (CAR) from [57]. The gap between the two solutions is computed as follows GAP = $\frac{\text{Sol}_{\text{CAR}} - \text{Sol}_{\text{Duplex}}}{\text{Sol}_{\text{CAR}}}$, where Sol_{CAR} and $\text{Sol}_{\text{Duplex}}$ are the solutions obtained using the CAR and the neurodynamic duplex, respectively. For the neurodynamical duplex, we take $\frac{\kappa_1}{\kappa_2} = 0.1$ for the first dynmical neural network and $\frac{\kappa_1}{\kappa_2} = 10.0$ for the second one.

6.2.6 . Uncertainty Sets with First Two Order Moments

A three-dimension shape optimization problem

We first consider a transportation problem involving the shifting of grain from a warehouse to a factory. The grain is transported within an open rectangular box, with dimensions of length x_1 meters, width x_2 meters, and height x_3 meters, as illustrated in Figure 6.2. The objective of the problem is to maximize the volume of the rectangular box, given by the product of its length, width, and height ($x_1 x_2 x_3$). However, two constraints must be satisfied. The first constraint relates to the floor area of the box, and the second constraint relates to the wall area. These constraints are necessary to ensure that the shape of the box aligns with the requirements of a given truck. In our analysis, we assume that the wall area A_{wall} and the floor area A_{floor} are random variables. We formulate our shape optimization problem as follows

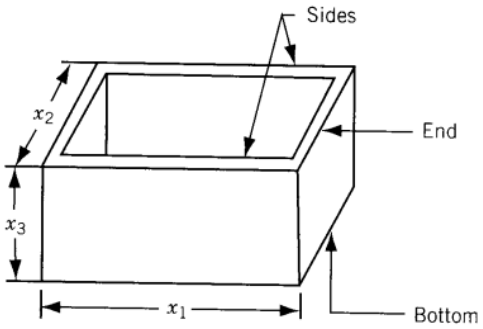

Figure 6.2: 3D-box shape [1]

Independent case			Dependent case		
Obj value	CPU Time	VS	Obj value	CPU Time	VS
0.296	0.43	0	0.298	0.46	0

Table 6.1: Results of solving problem (6.69) when $\mathcal{D} = \mathcal{D}^2$

$$\min_{x \in \mathbb{R}^3_{++}} \; x_1^{-1} x_2^{-1} x_3^{-1}, \tag{6.69}$$
$$\text{s.t} \quad \inf_{\mathcal{F} \in \mathcal{D}} \mathbb{P}\left(\frac{1}{A_{wall}}(2x_3x_2 + 2x_1x_3) \leq 1, \frac{1}{A_{floor}} x_1 x_2 \leq 1 \right) \geq 1 - \epsilon.$$

where $\mathcal{F}$ is the joint distribution for $\frac{1}{A_{wall}}$ and $\frac{1}{A_{floor}}$ and $\mathcal{D}$ is the uncertainty set for the probability distribution $\mathcal{F}$. We solve problem (6.69) when the uncertainty set is equal to $\mathcal{D}^2$ using the dynamical neural network (6.37)-(6.38). For the numerical experiments, we take the mean and the covariance describing the uncertainty sets for $\frac{1}{A_{wall}}$ $m_{wall} = 0.05$, $\sigma_{wall} = 0.01$, respectively and for $\frac{1}{A_{floor}}$ $m_{floor} = 0.5$, $\sigma_{floor} = 0.1$, respectively. We recapitulate the obtained results in Table 6.1. Columns one, two and three give the optimal value, the CPU time and the number of violated scenarios (VS) in the independent case, respectively. Columns four, five and six show the optimal value, the CPU time and the number VS in the dependent case, respectively. We observe that the dependent case is more conservative compared to the independent and that the dynamic neural network covers well the risk region in both cases. Figure 6.3 show the convergence of the state variables.

A generalized shape optimization problem

To further assess the performance of our dynamical neural network, we employed the multidimensional shape optimization problem with joint chance constraints from [89].

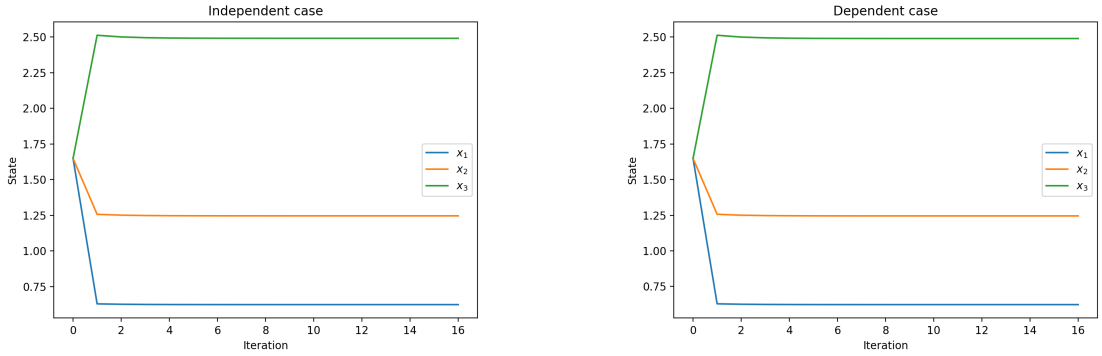

Figure 6.3: Transient behaviors of the state variables

$$\begin{aligned} &\min_{x\in\mathbb{R}^M_{++}} \prod_{i=1}^{m} x_i^{-1}, \\ &\text{s.t} \quad \inf_{\mathcal{F}\in\mathcal{D}} \mathbb{P}_{\mathcal{F}}\left(\sum_{j=1}^{m-1}\left(\frac{m-1}{A_{wall_j}}x_1 \prod_{i=1,i\neq j}^{m} x_i\right), \frac{1}{A_{floor}}\prod_{j=2}^{m} x_j \leq 1\right) \geq 1-\epsilon, \qquad (6.70) \\ &\quad \frac{1}{\gamma_{i,j}} x_i x_j^{-1} \leq 1, 1 \leq i \neq j \leq m. \end{aligned}$$

In our numerical experiments, we fixed the following parameters $\frac{1}{\gamma_{i,j}} = 0.5$ and $\epsilon = 0.15$. The inverse of floor's area ($\frac{1}{A_{floor}}$) and the inverse of wall area ($\frac{1}{A_{wall_j}}$) for each $j = 1, ..., m$ were considered as random variables. We test the robustness of the different approaches by creating 100 random samples of the variables $\frac{1}{A_{wall_j}}$ and $\frac{1}{A_{floor}}$. We then examine if the solutions meet the constraint of (6.70) for all 100 cases for the gaussian distribution for example. If the solutions are not feasible for a particular case, it is referred to as a violated scenario (VS).

We first solve (6.70) for $m = 5$ and when the uncertainty set is $\mathcal{D}^2$ in the independent case for different initial points, we observe that the dynamical neural network (6.37)-(6.38) converges to the same final value independently from the starting value as shown in Figure 6.4.

Now we solve (6.70) when we know the first-order moments of $\frac{1}{A_{floor}}$ and $\frac{1}{A_{wall_j}}$ for both the dependent and the independent case. We present the obtained results in Table 6.6. We observe again that the dependent case is more conservative compared to the independent one.

6.2.7 . Uncertainty Sets with Known First Order Moment and Nonnegative Support

A generalized shape optimization problem

We solve (6.70) when the uncertainty set is $\mathcal{D}^3$ for both the independent and the dependent case.

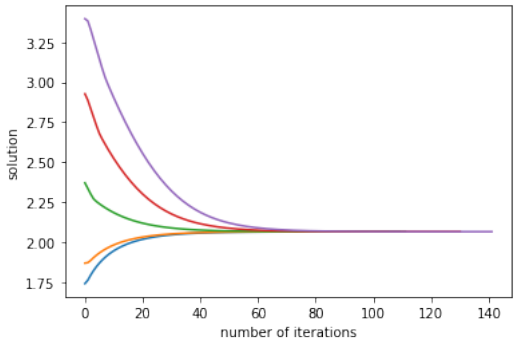

Figure 6.4: Convergence of the dynamical neural network (6.37)-(6.38) for different initial points for (6.70).

m	Independent case	CPU Time	Dependent case	CPU Time
3	1.03	1.05	1.30	1.39
5	2.09	5.11	2.15	5.20
10	14.79	4.83	5.04	15.10
15	7.76	47.80	7.99	58.04
20	10.68	97.72	10.87	100.91

Table 6.2: Results for different values of m

For the numerical experiments, we take $\epsilon = 0.2$. We solve problem (6.70) using the neurodynamic duplex in the dependent case. We recapitulate the obtained results in Table 6.3. Column one gives the number of variables m. Columns two, three and four give the objective value, the CPU time and the number of VS in the independent case, respectively. Columns five, six and seven give the objective value, the CPU time and the number of VS for the dependent case, respectively. We observe that the problem with dependent variables is more conservative. Nevertheless, the solution, in this case, covers well the risk area as the number of VS is equal to 0 for all the values of m.

Now we additionally solve problem (6.70) using the assumption that the random variables follow a normal distribution [132] for $m = 5$. In order to compare the solutions obtained with the stochastic and the robust approaches, we evaluate the robustness of the solutions for different hypotheses on the true distribution of the random parameters, i.e, the uniform distribution, the normal distribution, the log-normal distribution, the logistic distribution and Gamma distribution. The obtained results are presented in Table 6.4 which gives the number of violated scenarios for both the normal solutions and the robust ones and the objective value obtained by each solution. We can infer that the distributionally robust approaches are a conservative approximation of the stochastic programs. We observe that the solutions obtained by the nonnegative support are more conservative compared to the stochastic ones. Notice that the distributionally robust solutions are

m	Independent case			Dependent case		
	Obj value	CPU Time	VS	Obj value	CPU Time	VS
3	0.204	2.28	3	0.491	10.12	0
5	1.03	6.25	2	1.82	98.68	0
10	6.99	15.26	2	9.79	86.35	0
15	18.43	23.84	3	23.45	201.13	0
20	32.09	94.76	5	38.71	744.26	0
30	42.37	100.23	3	51.56	1155.42	0

Table 6.3: (6.70) for different values of m for $\mathcal{D} = \mathcal{D}^3$

more robust, i.e., the number of VS when the true distribution is the Logistic distribution is equal to 23 and 19 for the nonnegative support solutions and is equal to 0 for the robust solutions.

		Normal solutions		Robust solutions	
		Independent	Dependent	Independent	Dependent
Number of violated scenarios	Objective Value	0.86	0.99	2.43	4.14
	Uniform distribution	22	15	0	0
	Normal distribution	18	11	1	0
	Log-normal distribution	7	4	2	1
	Logistic distribution	23	19	0	0
	Gamma distribution	16	12	2	2

Table 6.4: Number of violated scenarios for the stochastic and the robust solutions

Maximizing the worst user signal-to-interference noise ratio

We consider the problem of maximizing the worst user signal-to-interference noise ratio (SINR) for Massive Multiple Input Multiple Output (MaMIMO) systems subject to antenna assignment and mul- tiuser interference constraints taken from [4] and given by

$$\max_{p \in \mathcal{K}_{++}} \min_{i \in \mathcal{U}} \frac{p_i |g_i^H g_i|^2}{\sum\limits_{j \in \mathcal{U}, j \neq i} p_j |g_i^H g_j|^2 + |\sigma_i|^2}, \tag{6.71}$$

$$\text{s.t} \quad P_{min} \leq p_i \leq P_{max}, \forall i \in \mathcal{U}, \tag{6.72}$$

where p_i is the power to be assigned for each user $i \in \mathcal{U}$. $g_i \in \mathbb{C}^{T \times 1}$, $g_i^H \in \mathbb{C}^{1 \times T}$ and σ_i^2 are the beam domain channel vector associated to user $i \in \mathcal{U}$, its Hermitian transpose and Additive White Gaussian Noise (AWGN), respectively.

Let $a_{ij} = |g_i^H g_j|^2 |g_i^H g_i|^{-2}$ and $b_i = |\sigma_i|^2 |g_i^H g_i|^{-2}$, we derive a geometric reformulation of (6.71)-(6.72)

$$\min_{p \in \mathcal{P}_{++}^K, w \in_{++}} \quad w^{-1}, \tag{6.73}$$

$$\text{s.t} \quad \sum_{j \in \mathcal{U}, j \neq i} a_{ij} p_j p_i^{-1} w + b_i p_i^{-1} w \leq 1, \forall i \in \mathcal{U}, \tag{6.74}$$

$$P_{min} \leq p_i \leq P_{max}, \forall i \in \mathcal{U}. \tag{6.75}$$

We assume that the coefficients a_{ij} and b_i are independent random variables and we propose the following optimization problem with individual and joint chance constraints

$$\begin{aligned} \min_{p \in \mathcal{P}_{++}^K, w \in_{++}} \quad & w^{-1}, \\ \text{s.t} \quad & \inf_{\mathcal{F}_i \in \mathcal{D}-i} \mathbb{P}_{\mathcal{F}_i} \left\{ \sum_{j \in \mathcal{U}, j \neq i} a_{ij} p_j p_i^{-1} w + b_i p_i^{-1} w \leq 1 \right\} \geq 1 - \epsilon_i, \forall i \in \mathcal{U}, \\ & P_{min} \leq p_i \leq P_{max}, \forall i \in \mathcal{U}. \end{aligned} \tag{POI}$$

and

$$\begin{aligned} \min_{p \in \mathcal{P}_{++}^K, w \in_{++}} \quad & w^{-1}, \\ \text{s.t} \quad & \inf_{\mathcal{F} \in \mathcal{D}} \mathbb{P}_{\mathcal{F}} \left\{ \sum_{j \in \mathcal{U}, j \neq i} a_{ij} p_j p_i^{-1} w + b_i p_i^{-1} w \leq 1, \forall i \in \mathcal{U} \right\} \geq 1 - \epsilon, \\ & P_{min} \leq p_i \leq P_{max}, \forall i \in \mathcal{U}. \end{aligned} \tag{POJ}$$

We assume that the uncertainty set for the distributionally robust problems (POI) and (POJ) is $\mathcal{D}^3$. We fix $\epsilon = 0.2$. We first solve problem (POJ) for $K = 10$. Figure 6.5 shows the convergence of the power variables. Next, we solve (POI) and (POJ) for different values of the number of users K. Table 6.5 presents the obtained results. Column one gives the number of users K. Columns two and three give the optimal value and the number of VS for (POI), respectively. Columns four and five show the optimal value and the number of VS for (POJ), respectively. As observed in the previous section, the use of joint constraints leads to a more conservative minimization problem but covers well the risk area compared to the problem with individual constraints since the number of VS is lower.

6.3 . Linear programs as a particular case

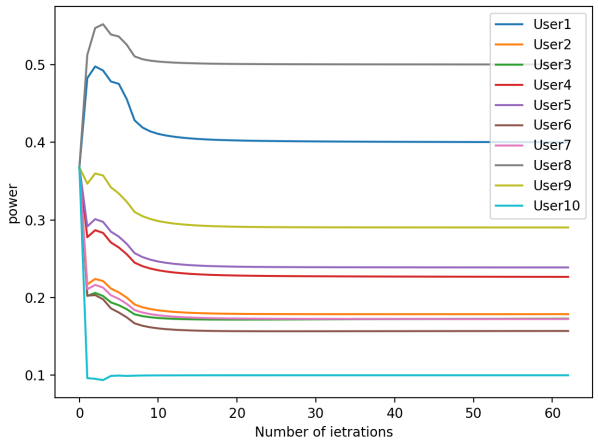

Figure 6.5: Convergence of the power variables

K	Individual constraints		Joint constraints	
	Obj value	VS	Obj value	VS
5	27.27	5	29.07	0
10	47.36	4	50.23	0
15	66.03	5	68.76	1
20	123.48	3	127.43	0

Table 6.5: Results for different values of K

We consider in this section the case of linear programs. The distributionally robust problem can be formulated as follows.

$$\min_{x\in_+^n} \sup_{\mathcal{F}_0\in\mathcal{D}_0} \mathbb{E}_{\mathcal{F}_0}\left[\tilde{\zeta_0}^T x\right], \tag{6.76}$$

$$\text{s.t} \quad \inf_{\mathcal{F}\in\mathcal{D}} \mathbb{P}_{\mathcal{F}}\left(\tilde{\zeta}_k x \leq \mathsf{b}_k, k=1,...,K\right) \geq \alpha. \tag{6.77}$$

where $\tilde{\zeta_0} \in^n$ is an uncertain parameter, $[\tilde{\zeta_1}, \tilde{\zeta_2}, ...\tilde{\zeta_K}]^T$ is a $K \times n$ set of pairwise independent random vectors in ^{n} and b $\in^K$ is a deterministic vector. We consider the case where the probability distribution $\mathcal{F}_0$ of $\tilde{\zeta_0}$ belongs to a certain uncertainty set $\mathcal{D}_0$ and the probability distributions $\mathcal{F}_k$ of $\tilde{\zeta}_k$, $k = 1, ..., K$ are not completely known and belong to $\mathcal{D}_k$. Thus, we take the worst-case where constraints (6.77) are jointly satisfied for all possible distributions in a given distributional uncertainty set $\mathcal{D}$ with a given probability level α. Based on the pairwise independence between the vectors $(\tilde{\zeta}_k)_{k\in\{1,..,K\}}$, we introduce

nonnegative auxiliary variables z_k, $k = 1, ..., K$ and rewrite the constraint (6.77) as

$$\begin{cases} \inf\limits_{\mathcal{F}_k \in \mathcal{D}_k} \mathbb{P}_{\mathcal{F}_k}\left(\tilde{\zeta}_k x \leq b_k\right) \geq \alpha^{z_k}, k = 1, ..., K \\ \sum\limits_{k=1}^{K} z_k = 1, \\ z_k \geq 0, k = 1, ..., K. \end{cases} \tag{6.78}$$

In the following subsections, we provide deterministic equivalent formulations for the optimization problem described in equations (6.76)-(6.77). To handle the uncertainty in the problem, we consider two moments-based uncertainty sets to define the sets $\mathcal{D}_k$, where $k = 1, ..., K$.

We begin by assuming that we have knowledge of the mean vector μ_k and the covariance matrix Σ_k associated with the random vector $\tilde{\zeta}_k^T$. Based on this information, we define the following uncertainty ser for each $k = 0, 1, ..., K$.
$\mathcal{D}_k^1(\mu_k, \Sigma_k) = \left\{ \mathcal{F}_k \;\middle|\; \begin{array}{r} \mathbb{E}[\tilde{\zeta}_k^T] = \mu_k \\ \mathbb{E}[(\tilde{\zeta}_k^T - \mu_k)(\tilde{\zeta}_k^T - \mu_k)^T] = \Sigma_k \end{array} \right\}$, where $\mathcal{F}_k$ is a probability distribution of $\tilde{\zeta}_k^T$. In the case where we have the mean vector μ_k and the covariance matrix Σ_k of $\tilde{\zeta}_k^T$ available, we can obtain a deterministic reformulation for the distributionally robust joint chance constraint (6.77) as follows [26].

$$\begin{cases} \mu_k^T x + \sqrt{\dfrac{\alpha^{z_k}}{1 - \alpha^{z_k}}} ||\Sigma_k^{\frac{1}{2}} x|| \leq b_k, k = 1, ..., K \\ \sum\limits_{k=1}^{K} z_k = 1, \\ z_k \geq 0, k = 1, .., K. \end{cases} \tag{6.79}$$

The deterministic equivalent problem for (6.76)-(6.77) under these assumptions is given by.

$$\min \mu_0^T x, \tag{6.80}$$

$$\text{s.t. } \mu_k^T x + \sqrt{\frac{\alpha^{z_k}}{1 - \alpha^{z_k}}} ||\Sigma_k^{\frac{1}{2}} x|| \leq b_k, k = 1, ..., K \tag{6.81}$$

$$\sum_{k=1}^{K} z_k = 1, x \geq 0, \tag{6.82}$$

$$z_k \geq 0, k = 1, .., K. \tag{6.83}$$

Lemma 6o. The function $z \mapsto \sqrt{\frac{\alpha^z}{1-\alpha^z}}$, with $0 < \alpha < 1$ is convex $\forall z > 0$.

Proof. Let $z > 0$ and $0 < \alpha < 1$, we have $\sqrt{\frac{\alpha^z}{1-\alpha^z}} = \exp\left\{\frac{1}{2}(z\log(\alpha) - \log(1 - \alpha^z))\right\}$. We have $z \mapsto \alpha^z$ is a convex function and the function $z \mapsto \log(1 - z)$ is non-increasing and concave, there follows that $z \mapsto \log(1-\alpha^z)$ is concave. We have then that $z \mapsto \frac{1}{2}(z\log(\alpha) -$

$\log(1-\alpha^z))$ is convex as an addition of two convex functions. Furthermore, $z \mapsto e^z$ is a non-increasing convex function we conclude then that $z \mapsto \exp\left\{\frac{1}{2}(z\log(\alpha) - \log(1-\alpha^z))\right\}$ is convex. The conclusion follows. □

Corollary 61. Problem (6.80)-(6.83) is biconvex on (x, z)

Now we consider the case where the mean of $\tilde{\zeta}k$ lies in an ellipsoid of size $\gamma k1 \geq 0$ with center μ_k and the covariance matrix of $\tilde{\zeta}_k$ lies in a positive semidefinite cone of center Σ_k. We define for every $k = 0, 1, ..., K$,
$\mathcal{D}_k^2(\mu_k, \Sigma_k) = \left\{ \mathcal{F}_k \;\middle|\; \begin{array}{r} (\mathbb{E}_{\mathcal{F}_k}[\tilde{\zeta}_k^T] - \mu_k)^T \Sigma_k^{-1} (\mathbb{E}_{\mathcal{F}_k}[\tilde{\zeta}_k^T] - \mu_k) \leq \gamma_{k1} \\ \text{COV}_{\mathcal{F}_k}(\tilde{\zeta}_k^T) \preceq \gamma_{k2}\Sigma_k \end{array} \right\}$, where $\gamma_{k2} \geq 0$ and $\text{COV}_{\mathcal{F}_k}$ is a covariance operator under probability distribution $\mathcal{F}_k$. The deterministic reformulation for the distributionally robust joint chance constraint (6.77) in this case where the mean lies in an ellipsoid and the covariance matrix lies in a positive semidefinite cone is provided in [106].

$$\begin{cases} \mu_k^T x + \left(\sqrt{\frac{\alpha^{z_k}}{1-\alpha^{z_k}}} \sqrt{\gamma_{k2}} + \sqrt{\gamma_{k1}} \right) ||\Sigma_k^{\frac{1}{2}} x|| \leq b_k, k = 1, ..., K \\ \sum_{k=1}^{K} z_k = 1, \\ z_k \geq 0, k = 1, .., K. \end{cases} \tag{6.84}$$

The objective function can be formulated as [89].

$$\min_{x \in \mathbb{R}^n_+} \mu_0^T x + \sqrt{\gamma_{01}} ||\Sigma_0^{\frac{1}{2}} x||. \tag{6.85}$$

The constraints set (6.84) is biconvex and the objective function (6.85) is convex.
To study the optimality conditions of the robust joint chance-constrained problem. We give the equivalent deterministic problem for each uncertainty set in a general form as follows.

$$\min\ f(x), \tag{6.86}$$
$$\text{s.t. } g_k(x, z) \leq 0, k = 1, ..., K, \tag{6.87}$$
$$h(z) \leq 0, \tag{6.88}$$
$$l(x) \leq 0, \tag{6.89}$$

where, $f(x) = \begin{cases} \mu_0^T x, & \text{if} \quad \mathcal{D}_k = \mathcal{D}_k^1 \\ \mu_0^T x + \sqrt{\gamma_{01}} ||\Sigma_0^{\frac{1}{2}} x||, & \text{if} \quad \mathcal{D}_k = \mathcal{D}_k^2 \end{cases}$, $h(z) = (\sum_{k=1}^{K} z_k - 1, 1 - \sum_{k=1}^{K} z_k, -z_1, -z_2, ..., z_K)^T$, $l(x) = -x$ and

$g_k(x, z) = \begin{cases} \mu_k^T x + \sqrt{\frac{\alpha^{z_k}}{1-\alpha^{z_k}}} ||\Sigma_k^{\frac{1}{2}} x|| - b_k, & \text{if} \quad \mathcal{D}_k = \mathcal{D}_k^1 \\ \mu_k^T x + (\sqrt{\frac{\alpha^{z_k}}{1-\alpha^{z_k}}} \sqrt{\gamma_{k2}} + \sqrt{\gamma_{k1}}) ||\Sigma_k^{\frac{1}{2}} x|| - b_k, & \text{if} \quad \mathcal{D}_k = \mathcal{D}_k^2 \end{cases}$.

Definition 22. Let $\mathcal{U}$ the feasible set of (6.86)-(6.89), $\mathcal{U}_x = \{z \mid g_k(x,z) \leq 0, h(z) \leq 0, k = 1,...,K\}$ and $\mathcal{U}_z = \{x \mid g_k(x,z) \leq 0, l(x) \leq 0, k = 1,...,K\}$. (x^*, z^*) is a partial optimum of (6.86)-(6.89) if $f(x^*) \leq f(x)$, $\forall x \in \mathcal{U}_{z^*}$.

Lemma 62. Let $\mathcal{U}$ the feasible set of (6.86)-(6.89) and $(x^*, z^*) \in \mathcal{U}$. If there exists $\beta^{(1)}$, $\beta^{(2)}$, γ and λ such that (x^*, z^*) verifies

$$\nabla_x f(x) + {\beta^{(1)}}^T \nabla_x g(x,z) + \lambda^T \nabla_x l(x) = 0, \tag{6.90}$$

$$\lambda \geq 0, \lambda^T l(x) = 0, \beta^{(1)} \geq 0, {\beta^{(1)}}^T g(x,z) = 0, \tag{6.91}$$

$${\beta^{(2)}}^T \nabla_z g(x,z) + \gamma^T \nabla_z h(z) = 0, \tag{6.92}$$

$$\beta^{(2)} \geq 0, {\beta^{(2)}}^T g(x,z) = 0, \gamma \geq 0, \gamma^T h(z) = 0, \tag{6.93}$$

where $g(x,z) = (g_1(x,z),...,g_K(x,z))$ and $(x)_+ = \max(0,x)$. Then (x^*, z^*) is a partial KKT point of (6.86)-(6.89).

Theorem 63. The partial KKT system (6.90)-(6.93) is equivalent to the following system

$$\nabla_x f(x) + \nabla_x g(x,z)^T (\beta^{(1)} + g(x,z))_+ + \nabla_x l(x)(\lambda + l(x))_+ = 0 \tag{6.94}$$

$$\nabla_z g(x,z)^T (\beta^{(2)} + g(x,z))_+ + \nabla_z h(z)^T (\gamma + h(z))_+ = 0 \tag{6.95}$$

$$(\beta^{(1)} + g(x,z))_+ - \beta^{(1)} = 0 \tag{6.96}$$

$$(\beta^{(2)} + g(x,z))_+ - \beta^{(2)} = 0 \tag{6.97}$$

$$(\lambda + l(x))_+ - \lambda = 0 \tag{6.98}$$

$$(\gamma + h(z))_+ - \gamma = 0 \tag{6.99}$$

Proof. By $(\beta^{(1)} + g(x,z))_+ = \beta^{(1)}$ and $(\lambda + l(x))_+ = \lambda$, we have

$$\Big(\nabla_x f(x) + \nabla_x g(x,z)^T (\beta^{(1)} + g(x,z))_+ + \nabla_x l(x)(\lambda + l(x))_+ = 0\Big)$$

is equivalent to

$$\Big(\nabla_x f(x) + {\beta^{(1)}}^T \nabla_x g(x,z) + \lambda^T \nabla_x l(x) = 0\Big)$$

We then obtain the equation (6.93) of the partial KKT system.

Furthermore, observe that

- $(\beta^{(1)} + g(x,z))_+ - \beta^{(1)} = 0$ if and only if $\beta^{(1)} \geq 0$, $g(x,z) \leq 0$ and ${\beta^{(1)}}^T g(x,z) = 0$,
- $(\lambda + l(x))_+ - \lambda = 0$ if and only if $\lambda \geq 0$, $l(x) \leq 0$ and $\lambda^T l(x) = 0$,

which leads to the equation (6.91) of the partial KKT system. We obtain the remaining equations following the same lines.
The converse part of the theorem is straightforward.

□

The following theorem gives the optimality conditions of problem (6.86)-(6.89) similarly to the previous Chapters.

Theorem 64. If partial Slater constraint qualification hold for (6.86)-(6.89) at (x^*, z^*), then (x^*, z^*) is a partial optimum of (6.86)-(6.89) if and only if (x^*, z^*) is a partial KKT point of (6.86)-(6.89). Furthermore, if $\beta^{(1)} = \beta^{(2)}$ then (x^*, z^*) is a KKT point of (6.86)-(6.89).

6.3.1 . A neurodynamic duplex

Based on the projection equations (6.47)-(6.49), we propose a duplex of two two-time-scale recurrent neural network models for solving (6.86)-(6.89)

$$\kappa_1 \frac{dx}{dt} = -\left(\nabla_x f(x) + \nabla_x g(x,z)^T(\beta + g(x,z))_+ + \nabla_x l(x)(\lambda + l(x))_+\right), \tag{6.100}$$

$$\kappa_2 \frac{dz}{dt} = -\left(\nabla_z g(x,z)^T(\beta + g(x,z))_+ + \nabla_z h(z)^T(\gamma + h(z))_+\right), \tag{6.101}$$

$$\kappa_2 \frac{d\beta}{dt} = -\beta + (\beta + g(x,z))_+, \tag{6.102}$$

$$\kappa_2 \frac{d\lambda}{dt} = -\lambda + (\lambda + l(x))_+. \tag{6.103}$$

$$\kappa_2 \frac{d\gamma}{dt} = -\gamma + (\gamma + h(z))_+. \tag{6.104}$$

where $(x, z, \beta, \gamma, \lambda)$ are now time-dependent variables and κ_1 and κ_2 are two time scaling constants with $\kappa_1 \neq \kappa_2$. We consider duplex of two two-time-scale recurrent neural network (6.100)-(6.104) for solving (6.86)-(6.89) one with $\kappa_1 > \kappa_2$ and the second with $\kappa_1 < \kappa_2$ as shown in Figure 6.6. The zoom on RNN1 shows the circuit implementation of a single two-timescale recurrent neural network (6.100)-(6.104).

Theorem 65. $(x^*, z^*, \beta^*, \gamma^*, \lambda^*)$ is an equilibrium point of (6.100)-(6.104) if and only if (x^*, z^*) is a KKT point of (6.86)-(6.89) and β^*, γ^* and λ^* are the associated Lagrange variables.

Proof. $(x^*, z^*, \beta^*, \gamma^*, \lambda^*)$ is an equilibrium point of (6.100)-(6.104) if and only if $\frac{dx}{dt} = 0$, $\frac{dz}{dt} = 0$, $\frac{d\beta}{dt} = 0$, $\frac{d\lambda}{dt} = 0$ and $\frac{d\gamma}{dt} = 0$, we then obtain equations (6.94)-(6.99). By Theorems 63 and 64. □

First, the state variables of the neurodynamic models are initialized. Then, each model undergoes a precise local search based on its dynamics for the optimization process. Once all neurodynamic models have converged to their equilibria, the initial states of the recurrent neural networks are optimized using the updating rule of particle swarm optimization (PSO). We denote $y_i = (y_{i1}, ..., y_{in})^T$ the position of the i^{th} particle and $v_i = (v_{i1}, ..., v_{in})^T$ its velocity. The inertia weight $w \in [0, 1]$ determines the degree to which the particle's previous velocity is retained. The best previous position yielding the maximum fitness value for the i^{th} particle is denoted as $\tilde{y}_i = (\tilde{y}_{i1}, ..., \tilde{y}_{in})^T$, and the best position yielding the maximum fitness value in the swarm is represented by $\hat{y} = (\hat{y}_1, ..., \hat{y}_n)^T$. The initial state of each neurodynamic model is updated using the PSO updating rule given by [34], i.e,

$$v_i(j+1) = wv_i(j) + c_1 r_1(\tilde{y}_i - y_i(j)) + c_2 r_2(\hat{y}_i - y_i(j)), \tag{6.105}$$

$$y_i(j+1) = y_i(j) + v_i(j+1). \tag{6.106}$$

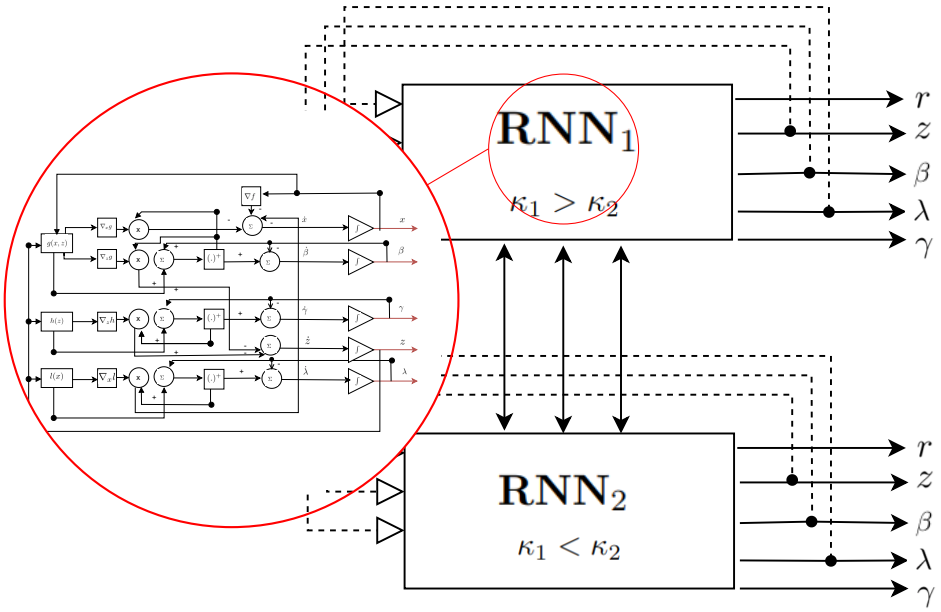

Figure 6.6: A block diagram depicting a duplex neurodynamic system with a two-timescale configuration

where the iterative index is represented by j, while the two weighting parameters are denoted as c_1 and c_2 and r_1 and r_2 represent two random values drawn from the interval $[0, 1]$.

The diversity of initial neuronal states plays a critical role in achieving global optimization. Introducing a mutation operator, which generates a random $y_i(j+1)$, can enhance the diversity of initial neuronal states. To quantify the diversity of these states, we use the following function

$$d = \frac{1}{n}\sum_{i=1}^{n}\|y_i(j+1) - \hat{y}(j)\|. \tag{6.107}$$

We use wavelet mutation operator from [86] and performing for the i^{th} particle if $d < \zeta$ as follows

$$y_i(j+1) = \begin{cases} y_i(j) + \mu(h_i - y_i(j)) & , \rho > 0 \\ y_i(j) + \mu(y_i(j) - l_i) & , \rho < 0 \end{cases} \tag{6.108}$$

where h_i and l_i are the upper and the lower bound for y_i, respectively. $\zeta > 0$ is a given threshold and ρ is defined using a wavelet function

$$\rho = \frac{1}{\sqrt{a}} e^{-\frac{\phi}{2a}} \cos(5\frac{\phi}{a}) \tag{6.109}$$

As the value of ρ approaches 1, the mutated element of the particle will move towards the maximum value of $y_i(j+1)$, whereas approaching -1, the mutated element goes towards the minimum value of $x_i(j+1)$. The magnitude of $|\rho|$ determines the size of the search space for $x_i(j+1)$, with larger values indicating a wider search space. Conversely, smaller values of $|\mu|$ lead to a smaller search space for fine-tuning. To achieve fine-tuning, the value of the dilation parameter a is adjusted based on the current number of iterations j relative to the total number of iterations T. Specifically, a is a function of j/T, we take $a = e^{10\frac{j}{T}}$. We generate ϕ randomly from $[-2.5a, 2.5a]$.

The algorithm details are given in Algorithm 3 where $y = (x, z, \beta, \gamma)$

Algorithm 3 The neurodynamic duplex

- Let $y_1(0)$ and $y_2(0)$ be randomly generated in the feasible region.
- Let $\tilde{y}(0) = \hat{y}(0) = y(0)$ the initial best previous position and best position, respectively.
-Set the convergence error ϵ.
while $||y(j+1) - y(j)|| \geq \epsilon$ **do**
  Compute the equilibrium points $\bar{y}_1(j)$ and $\bar{y}_2(j)$ of RNN$_1$ and RNN$_2$ based on (6.28)-(4.96).
  if $f(\bar{x}_1(j)) < f(\tilde{x}(j))$ **then**
    $\tilde{y}(j+1) = \bar{y}_1(j)$
  else
    $\tilde{y}(j+1) = \tilde{y}(j)$
  end if
  if $f(\bar{x}_2(j)) < f(\tilde{x}(j))$ **then**
    $\tilde{y}(j+1) = \bar{y}_2(j)$
  else
    $\tilde{y}(j+1) = \tilde{y}(j)$
  end if
  if $f(\tilde{x}(j)) < f(\hat{x}(j))$ **then**
    $\hat{y}(j+1) = \tilde{y}(j+1)$
  else
    $\hat{y}(j+1) = \hat{y}(j)$
  end if
  Compute the value of $y(j+1)$ following (6.105)-(6.106).
  if $d < \zeta$ **then**
    Perform the wavelet mutation (6.108).
  end if
  j=j+1
end while

6.3.2 . Convergence Analysis

Lemma 66. [137] Suppose that the objective function f is measurable, and the feasible region $\mathcal{U}$ is a measurable subset, and for any Borel subset $\mathcal{B}$ of $\mathcal{U}$ with positive Lebesgue measure we have $\prod_{k=1}^{\infty}(1-\mathbb{P}_k(\mathcal{B}))=0$. Let $\{y(k)\}_{k=1}^{\infty}$ be a sequence generated by a stochastic optimization algorithm. If $\{f(y(k))\}_{k=1}^{\infty}$ is a nonincreasing sequence, then it converges with probability one to the set of global optimal solutions.

Theorem 67. If the state of the following neurodynamic model with a single timescale,

$$\kappa\frac{dx}{dt}=-x+\left(x-(\nabla_x f(x)+\beta^T\nabla_x g(x,z))\right)_+, \tag{6.110}$$
$$\kappa\frac{dz}{dt}=-z+(z-(\beta^T\nabla_z g(x,z)+\gamma^T\nabla_z h(z)))_+, \tag{6.111}$$
$$\kappa\frac{d\beta}{dt}=-\beta+(\beta+g(x,z))_+, \tag{6.112}$$
$$\kappa\frac{d\lambda}{dt}=-\lambda+(\lambda+l(x))_+. \tag{6.113}$$
$$\kappa\frac{d\gamma}{dt}=-\gamma+(\gamma+h(z))_+. \tag{6.114}$$

converges to an equilibrium point, then the state of a neurodynamic model with two timescales, as described by equations (6.100)-(6.104) globally converges to a partial optimum of problem (6.86)-(6.89).

Proof. The Lagrangian function of problem (6.86)-(6.89) is given by

$$\mathcal{L}(x,z,\beta,\lambda,\gamma)=f(x)+\beta^T g(x,z)+\gamma^T l(x)+\lambda^T h(z). \tag{6.115}$$

As an equilibrium point of (6.110)-(6.114) corresponds to a KKT point $(x^*,z^*,\beta^*,\lambda^*,\gamma^*)$ of (6.86)-(6.89) [142] verifying

$$\nabla_x\mathcal{L}(x^*,z^*,\beta^*,\lambda^*,\gamma^*)=0, \tag{6.116}$$
$$\nabla_z\mathcal{L}(x^*,z^*,\beta^*,\lambda^*,\gamma^*)=0, \tag{6.117}$$
$$\nabla_\beta\mathcal{L}(x^*,z^*,\beta^*,\lambda^*,\gamma^*)=0, \tag{6.118}$$
$$\nabla_\gamma\mathcal{L}(x^*,z^*,\beta^*,\lambda^*,\gamma^*)=0, \tag{6.119}$$
$$\nabla_\lambda\mathcal{L}(x^*,z^*,\beta^*,\lambda^*,\gamma^*)=0, \tag{6.120}$$
$$\beta^{*T}g(x^*,z^*)=0,\beta^*\geq 0, \tag{6.121}$$
$$\gamma^{*T}l(x^*)=0,\gamma^*\geq 0. \tag{6.122}$$
$$\lambda^{*T}h(z^*)=0,\lambda^*\geq 0. \tag{6.123}$$

We fix x^* and take $z\in\mathcal{U}_x^*$, problem (6.86)-(6.89) becomes convex, we have then

$$\mathcal{L}(x^*,z^*,\beta^*,\lambda^*,\gamma^*)\leq\mathcal{L}(x^*,z,\beta^*,\lambda^*,\gamma^*), \tag{6.124}$$

which leads to

$$f(x^*)+\beta^{*T}g(x^*,z^*)+\lambda^{*T}h(z^*)\leq f(x^*)+\beta^{*T}g(x^*,z)+\lambda^{*T}h(z). \tag{6.125}$$

As $\lambda^{*T} l(x) \leq \lambda^{*T} l(x^*) = 0$, $\gamma^{*T} h(z) \leq \gamma^{*T} h(z^*) = 0$ and $\beta^{*T} g(x^*, z) \leq \beta^{*T} g(x^*, z^*) = 0$ from the KKT conditions, then $f(x^*) \leq f(x)$ and this for every $z \in \mathcal{U}_x^*$. By Definition 22, we have that x^* is a partial optimum of (6.86)-(6.89). □

Theorem 68. If the state of the two-timescale neurodynamic model (6.100)-(6.104) converges to a partial optimum and the initial states and time constants of the two neurodynamic models are sufficiently different. Then the duplex of two two-timescale neural networks in Figure 6.6 system is globally convergent to a global optimal solution of problem (6.86)-(6.89) with probability one.

Proof. By Theorem 58, the two-timescale neurodynamic models RNN_1 and RNN_2 are proven to converge to a partial optimum. From Algorithm 2, the solution sequence is generated as follows

$$\begin{cases} \hat{y}(j+1) & = \tilde{y}(j+1) \text{ if } f(\tilde{x}(j)) < f(\hat{x}(j)), \\ \hat{y}(j+1) & = \hat{y}(j) \text{ else.} \end{cases}$$

We observe that the generated solution sequence $\{f(\hat{y}(j))\}_{j=1}^{\infty}$ is monotonically increasing.

Let $\mathcal{M}_{i,j}$ be the supporting set of the initial state of RNN_i at iteration j. As indicated by equation (6.108), the mutation operation ensures that the initial states of the RNNs are forced to be in the feasible region $\mathcal{U}$. Hence, for every iteration index $J \geq 1$, the supporting sets fulfill the following condition

$$\mathcal{U} \subseteq \mathcal{M} = \bigcup_{j=1}^{J} \bigcup_{i=1}^{2} \mathcal{M}_{i,j}. \tag{6.126}$$

We have then $v(\mathcal{U}) = v(\mathcal{M}) > 0$. By Lemma 66, we have

$$\lim_{j->\infty} \mathbb{P}(\hat{y}(j) \in \Phi) = 1 \tag{6.127}$$

where Φ is the set of the global optimal solutions of (6.86)-(6.89). The conclusion follows. □

6.3.3 . Numerical experiments

To evaluate the performances of our approach, we consider a standard profit maximization problem. *A manufacturing firm produces n products with N different machines. The times required to manufacture each unit are random variables. The mean vector μ_j and the covariance matrix Σ_j describing the uncertainty sets of the time vector $t_j = \{t_{ij}\}_{1 \leq i \leq n}$, where t_{ij} is the time required to manufacture one unit of each of product i using machine j and the daily capacity of each machine j given by b_j are given. The objective of the study is to determine the daily number of units to be manufactured for each product without exceeding the available*

machining times. We write our robust joint chance-constrained maximization problem as follows.

$$\min \sup_{\mathcal{F}_0 \in \mathcal{D}_0} -\mathbb{E}\left[\tilde{c}^T x\right], \tag{6.128}$$

$$\text{s.t. } \inf_{\mathcal{F} \in \mathcal{D}} \mathbb{P}\left(\sum_{i=1}^{n} t_{ij} x_i \leq b_j, j = 1, ..., \mathsf{N}\right) \geq p, \tag{6.129}$$

$$x \geq 0, \tag{6.130}$$

where vector $\tilde{c}$ is a random variable and corresponds to the profit per unit for each product, t_{ij} is the time required to manufacture one unit of product i using machine j, b_j is the time capacity of machine j, p is a given probability level, $\mathcal{D}_0$ is an uncertainty set for the distribution $\mathcal{F}_0$ of $\tilde{c}$ and $\mathcal{D}$ is an uncertainty set for the distribution $\mathcal{F}$ of the random variables.

All the algorithms in this Section are implemented in Python. We run our algorithms on Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz. The random instances are generated with numpy.random, and we solve the ODE systems with solve_ivp of scipy.integrate. The deterministic equivalent programs are solved with the package gekko and the gradients and partial derivatives are computed with autograd.grad and autograd.jacobian. For the following numerical experiments, the values of μ_j and $\bar{c}$ the mean of $\tilde{c}$ are uniformly generated in $[2.0, 4.0]$, the components of the matrix Σ_j are uniformly drawn in the interval $[1.0, 3.0]$ and we generate the values of b_j uniformly in $[50.0, 60.0]$, $\gamma_{k1} = 5$ and $\gamma_{k2} = 5$.

The resulting deterministic equivalent problems of (6.128)-(6.130), where the uncertainty sets are $\mathcal{D}^1$ and $\mathcal{D}^2$ are given respectively by

$$\min \bar{c}^T x, \tag{6.131}$$

$$\text{s.t. } \mu_j^T x + \sqrt{\frac{p^{z_j}}{1 - p^{z_j}}} ||\Sigma_j^{\frac{1}{2}} x|| \leq b_j, j = 1, ..., N, \tag{6.132}$$

$$\sum_{j=1}^{N} z_j = 1, \tag{6.133}$$

$$x \geq 0, z_j \geq 0, j = 1, ..., N, \tag{6.134}$$

and

$$\min \bar{c}^T x, \tag{6.135}$$

$$\text{s.t. } \mu_j^T x + \left(\sqrt{\frac{p^{z_j}}{1 - p^{z_j}}} \sqrt{\gamma_{k2}} + \sqrt{\gamma_{k1}}\right) ||\Sigma_j^{\frac{1}{2}} x|| \leq b_j, j = 1, ..., N, \tag{6.136}$$

$$\sum_{j=1}^{N} z_j = 1, \tag{6.137}$$

$$x \geq 0, z_j \geq 0, j = 1, ..., N, \tag{6.138}$$

The neurodynamic duplex vs. convex approximations

Cheng et al. [26] propose two convex approximations to solve problem (6.128)-(6.130). A linear approximation that gives an upper bound to the minimization problem and a tangent approximations that leads to a lower bound. In this first subsection, we compare the objective value obtained using the neurodynamic duplex with those obtained using the linear and the tangent approximations. We compute the gap between the two bounds and the global optimum given by the neurodynamic duplex by GAP = $\frac{\text{Bound}_{\text{lower, upper}} - \text{ND}}{\text{Bound}_{\text{lower, upper}}}$, where $\text{Bound}_{\text{lower}}$ is the value of the lower bound, $\text{Bound}_{\text{upper}}$ is the value of the upper bound, and ND is the value obtained using the neurodynamic duplex. We recapitulate the obtained results in Table 6.6. Column one gives the value of the confidence parameter p. Column two gives the final value of the neurodynamic duplex. Columns three and four show the lower bound and its gap with the neurodynamic duplex, respectively. Finally, columns five and six present the upper bound and the gap with the neurodynamic approach. We observe that the final value obtained with the dynamical duplex remains between the two bounds for the different values of p with gaps less than 0.5%, demonstrating that the neurodynamic approach effectively converges to the global optimum. Moreover, we remark that as p increases the value of the objective function increase which is coherent since lower values of p induce larger risk area.

p	Neurodynamic duplex	Tangent approximation		Linear approximation	
	Obj value	Obj value	GAP	Obj value	GAP
0.95	-36.25	-36.41	0.43%	-36.20	-0.13%
0.9	-40.48	-40.51	0.07%	-40.46	-0.04%
0.8	-45.30	-45.41	0.24%	-45.22	-0.17%
0.7	-47.31	-47.38	0.14%	-47.28	-0.06%
0.6	-48.09	-48.13	0.08%	-48.07	-0.06%

Table 6.6: Results for different values of p for $\mathcal{D}^1(\mu, \Sigma)$

n	N	Neurodynamic duplex	Tangent approximation		Linear approximation	
		Obj value	Obj value	GAP	Obj value	GAP
7	4	-22.86	-22.97	0.47%	-22.77	-0.39%
10	5	-22.51	-22.65	0.61%	-22.44	-0.31%
15	10	-21.36	-21.61	1.15%	-20.82	-2.59%
20	15	-21.28	-21.78	2.29%	-20.93	-1.67%
25	20	-19.79	-20.78	4.67%	-19.01	-4.10%

Table 6.7: Results for different values of n and N for $\mathcal{D}^1(\mu, \Sigma)$

The distributionally robust optimization approach vs. stochastic optimization approaches

To evaluate the robustness of the proposed duplex for the two uncertainty sets $\mathcal{D}^1$ and $\mathcal{D}^2$, we additionally solve problem (6.128)-(6.130) when the random variables follow uniform and normal distributions and $p = 0.95$. We compare the solution of our proposed distributionally robust approach with the solution of the stochastic programming approach. We generate 100 instances for $(t_{ij})_{1\leq I\leq n, 1\leq j\leq N}$ using the mean vectors and the covariance matrix when the true distribution of the stochastic variables is one of the five following distributions: uniform distribution, normal distribution, log-normal distribution, logistic distribution and Gamma distribution. We calculate the number of times when the constraints were violated over the 100 generated scenarios for each stochastic and robust solutions. Table 6.8 recapitulates the obtained results, where column one gives the true distribution, columns two, three, four and five give the number of violated scenarios for the solution obtained using the uniform approach, the normal approach, the first robust approach and the second robust approach, respectively. The relative expected profit is computed relatively to the value achieved by the solution of the stochastic program with uniform distribution.

We observe that the distributionally robust approaches are more conservative compared to the stochastic approaches. We invest between 4.3% and 12.2% of the expected profit in order to ensure the joint constraint. In fact, the average number of violated scenarios for the robust approaches are 0 while the numbers of violated scenarios for the stochastic solutions are significant, i.e., when Gamma is the true distribution of the random variables, the average number of the violated scenarios are 24 and 9 for the uniform and the normal solutions, respectively.

		Stochastic solutions		Robust solutions	
		Uniform	Normal	$\mathcal{D}^1(\mu, \Sigma)$	$\mathcal{D}^2(\mu, \Sigma)$
	Relative expected profit	-0%	-0.5%	-4.3%	-12.2%
Number of violated scenarios	Uniform distribution	2	0	0	0
	Normal distribution	8	5	0	0
	Log-normal distribution	15	6	0	0
	Logistic distribution	23	5	0	0
	Gamma distribution	24	9	0	0

Table 6.8: Number of violated scenarios for the stochastic and the robust solutions

7 - Conclusions and Perspectives

7.1 . Conclusions

Dynamical recurrent neural networks have emerged as a powerful tool for tackling optimization problems, showcasing notable advancements in both theory and applications within optimization theory. However, despite these significant strides, several important challenges remain unresolved, creating an appealing avenue for further research and exploration in neurodynamic optimization. This thesis has been dedicated to investigating the practical application of recurrent dynamical neural networks in solving various geometric optimization problems. The dissertation's contributions encompass the development of novel theoretical frameworks, algorithmic techniques, and computational methodologies that effectively incorporate chance-constrained geometric optimization. Additionally, the research conducted in this thesis extends to practical applications across diverse domains, including telecommunications, engineering, transportation, and shape optimization. Through investigating these real-world problem scenarios, valuable insights and solutions have been derived, facilitating informed decision-making processes.

Chapter 4 of this thesis is dedicated to the exploration of geometric programs with joint chance constraints. We focus specifically on the case where the stochastic parameters are independently distributed according to a normal distribution. To address this problem, we begin by deriving a biconvex deterministic reformulation of the initial chance-constrained problem through the application of a standard variable transformation technique. Next, we review existing state-of-the-art methods that rely on convex approximation. In contrast, we propose a novel neurodynamic approach that solves the stochastic program without the need for any approximations. We establish the stability and global convergence of the proposed neurodynamic approach using Lyapunov analysis. To validate the effectiveness of our approach, we conduct numerical experiments that demonstrate its robustness and superior performance compared to a sequential algorithm. Furthermore, we extend our research to consider the scenario of joint rectangular geometric chance-constrained programs. In this case, we obtain the biconvex equivalent formulation by employing the log transformation and leveraging the arithmetic-geometric mean inequality. In the final part of the chapter, we present a real-world application of our proposed method. Specifically, we apply it to solve a problem involving the maximization of signal-to-interference noise ratio for massive multiple-input, multiple-output (MIMO) systems.

In Chapter 5, we introduced the copula theory as an efficient tool to model chance-constrained optimization problems with dependent stochastic variables. We first consider the case of linear programs. We propose a neurodynamic approach to solve the equivalent deterministic program based on the optimality conditions given by the partial KKT system. We apply the proposed algorithm to solve a profit maximization problem. Further, we extend our approach to solve geometric-dependent chance-constrained

programs. The deterministic equivalent was obtained using an Archimedian copula combined with a log transformation. A generalized shape optimization problem was solved using the neurodynamical approach with different elliptical distributions.

Chapter 6 of this thesis focuses on distributionally robust chance-constrained optimization. Initially, we explore the linear case and consider two different uncertainty sets based on moments. For each uncertainty set, we derive the corresponding deterministic equivalent formulation. To tackle the obtained equivalents, we propose a duplex consisting of two two-timescale recurrent neural networks. This novel approach demonstrates convergence almost surely to a global optimum, providing a robust solution method for distributionally robust chance-constrained optimization problems in the linear case. In addition to the linear case, we extend our investigation to geometric programs. We propose two uncertainty sets: one with known first-order moments and the other with a known first-order moment and nonnegative support. We further consider the scenarios of independent and dependent random variables, resulting in four different deterministic programs. For the first three programs, where the obtained equivalents are convex, we utilize a single timescale dynamical neural network to solve them efficiently. However, for the fourth program, biconvex, we adapt the neurodynamic duplex approach to address its complexity and find the optimal solution effectively. By incorporating distributionally robust techniques and leveraging the power of recurrent neural networks, this chapter contributes to advancing the field of chance-constrained optimization. The proposed methodologies offer robustness and efficiency in solving linear and geometric distributionally robust chance-constrained programs. The convergence properties and adaptability of the neurodynamic duplex highlight its effectiveness in finding global optima for complex optimization problems.

In conclusion, the outcomes obtained in this dissertation not only enrich the existing knowledge base but also offer fresh perspectives and practical tools and methodologies for addressing geometric chance-constrained problems in real-world settings. The research presented herein contributes to advancing the field, paving the way for further exploration and application of recurrent dynamical neural networks in optimization contexts.

7.2 . Perspectives

In this dissertation, we explore the research landscape of geometric chance-constrained problems and propose novel neurodynamic algorithms to tackle these optimization challenges. While our work contributes valuable insights and algorithmic solutions, there are still open issues and future directions that warrant further investigation in the application of dynamical neural networks to chance-constrained optimization.

One important area for future research is the development of more efficient and scalable neurodynamic algorithms for solving large-scale chance-constrained optimization problems. The size and complexity of real-world problems often pose computational challenges, and exploring techniques to enhance the computational efficiency of neural

network-based approaches can greatly improve their applicability and practicality.

Furthermore, the exploration of alternative neural network architectures and learning paradigms could yield further improvements in addressing chance-constrained optimization problems. For instance, investigating recurrent neural networks with attention mechanisms or reinforcement learning-based approaches can offer new perspectives and potentially overcome limitations associated with traditional neurodynamic algorithms.

Moreover, the theoretical analysis of neurodynamic algorithms applied to chance-constrained optimization problems is an important research direction. Examining the convergence properties, stability guarantees, and robustness of these algorithms under different problem settings and assumptions can provide a deeper understanding of their behavior and facilitate their reliable application in practical scenarios.

Additionally, exploring the combination of dynamical neural networks with other optimization techniques and frameworks, such as evolutionary algorithms, swarm intelligence, or metaheuristics, can lead to hybrid approaches that leverage the strengths of different methods and offer improved performance in solving chance-constrained optimization problems.

Finally, we must highlight that the efficiency and quality of the developed algorithms can be improved through further research and development. One approach is to implement ODE solvers based on artificial intelligence techniques. Neural networks or reinforcement learning can potentially enhance the speed and accuracy of solving the dynamical differential system. These AI techniques can learn patterns and optimize the solution process, leading to faster and more precise results.

Additionally, other optimization techniques, such as parallel computing, GPU acceleration, or distributed computing, can accelerate the algorithm's execution time further. These techniques leverage hardware advancements to process computations in parallel, reducing the overall time required for solving the system.

By exploring these avenues of research and development, the algorithm can be made more efficient, allowing for faster and more effective solutions to chance-constrained geometric optimization problems.

Bibliography

[1] *Geometric Programming*, chapter 8, pages 492–543. John Wiley and Sons, Ltd, 2009. ISBN 9780470549124. doi: https://doi.org/10.1002/9780470549124.ch8.

[2] *Machine Learning and Wireless Communications*, page 182–211, 2022. doi: 10.1017/9781108966559.009.

[3] L. Adam, M. Branda, H. Heitsch, and R. Henrion. Solving joint chance constrained problems using regularization and benders' decomposition. *Annals of Operations Research*, 292(2):683–709, 2018. doi: 10.1007/s10479-018-3091-9.

[4] P. Adasme and A. Lisser. A stochastic geometric programming approach for power allocation in wireless networks. *Wireless Networks*, 2023. doi: 10.1007/s11276-023-03295-8.

[5] P. Adasme, I. Soto, E. S. Juan, F. Seguel, and A. D. Firoozabadi. Maximizing signal to interference noise ratio for massive mimo: A mathematical programming approach. In *2020 South American Colloquium on Visible Light Communications (SACVC)*, pages 1–6, 2020. doi: 10.1109/SACVLC50805.2020.9129889.

[6] L. Aliabadi, R. Heidari, R. Yazdanparast, and F. Jolai. In *Conference: 13th International Conference on Industrial Engineering (IIEC 2017)At: Babolsar, Iran*, 02 2017.

[7] K. M. Ang. A constrained teaching-learning-based optimization with modified learning phases for constrained optimization. *Journal of Advanced Research in Dynamical and Control Systems*, 12(SP4):1442–1456, 2020. doi: 10.5373/jardcs/v12sp4/20201623.

[8] M. Avriel and D. J. Wilde. Stochastic geometric programming. *Proceedings of the Princeton Symposium on Mathematical Programming*, page 73–92, 1970. doi: 10.1515/9781400869930-007.

[9] M. Avriel and A. C. Williams. Complementary geometric programming. *SIAM Journal on Applied Mathematics*, 19(1):125–141, 1970. ISSN 00361399. URL http://www.jstor.org/stable/2099336.

[10] M. Avriel and A. C. Williams. An extension of geometric programming with applications in engineering optimization. *Journal of Engineering Mathematics*, 5(2):187–194, 1971. doi: 10.1007/bf01535411.

[11] X. Bai, J. Sun, and X. Zheng. An augmented lagrangian decomposition method for chance-constrained optimization problems. *INFORMS Journal on Computing*, 33(3):1056–1069, 2021. doi: 10.1287/ijoc.2020.1001. URL https://doi.org/10.1287/ijoc.2020.1001.

[12] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998. ISSN 0364765X, 15265471. URL http://www.jstor.org/stable/3690632.

[13] J. BENDERS. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962/63. URL http://eudml.org/doc/131533.

[14] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004. doi: 10.1287/opre.1030.0065. URL https://doi.org/10.1287/opre.1030.0065.

[15] L. Blackmore, M. Ono, A. Bektassov, and B. Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *Robotics, IEEE Transactions on*, 26:502 – 517, 07 2010. doi: 10.1109/TRO.2010.2044948.

[16] D. L. Bricker and J. Rajgopal. Yet another geometric programming dual algorithm. *Operations Research Letters*, 2(4):177–180, 1983. ISSN 0167-6377. doi: https://doi.org/10.1016/0167-6377(83)90051-2. URL https://www.sciencedirect.com/science/article/pii/0167637783900512.

[17] P. Bürgisser, Y. Li, H. Nieuwboer, and M. Walter. Interior-point methods for unconstrained geometric programming and scaling problems. *ArXiv*, abs/2008.12110, 2020.

[18] V. Chandrasekaran and P. Shah. Relative entropy relaxations for signomial optimization. *SIAM Journal on Optimization*, 26(2):1147–1173, 2016. doi: 10.1137/140988978. URL https://doi.org/10.1137/140988978.

[19] A. Charnes and W. W. Cooper. Chance-constrained programming. *Management Science*, 6(1):73–79, 1959. URL https://EconPapers.repec.org/RePEc:inm:ormnsc:v:6:y:1959:i:1:p:73-79.

[20] A. Charnes and W. W. Cooper. Chance constraints and normal deviates. *Journal of the American Statistical Association*, 57(297):134–148, 1962. doi: 10.1080/01621459.1962.10482155. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1962.10482155.

[21] A. Charnes and W. W. Cooper. Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations Research*, 11(1):18–39, 1963. doi: 10.1287/opre.11.1.18. URL https://doi.org/10.1287/opre.11.1.18.

[22] A. Charnes, W. W. Cooper, and G. H. Symonds. Cost horizons and certainty equivalents: An approach to stochastic programming of heating oil. *Management Science*, 4(3):235–263, 1958. doi: 10.1287/mnsc.4.3.235. URL https://doi.org/10.1287/mnsc.4.3.235.

[23] W. Chen, M. Sim, J. Sun, and C.-P. Teo. From cvar to uncertainty set: Implications in joint chance-constrained optimization. *Operations Research*, 58(2):470–485, 2010. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/40605930.

[24] Z. Chen, S. Peng, and A. Lisser. A sparse chance constrained portfolio selection model with multiple constraints. *Journal of Global Optimization*, 77(4):825–852, 2020. doi: 10.1007/s10898-020-00901-3.

[25] J. Cheng and A. Lisser. A second-order cone programming approach for linear programs with joint probabilistic constraints. *Operations Research Letters*, 40(5):325–328, 2012. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2012.06.008. URL https://www.sciencedirect.com/science/article/pii/S0167637712000831.

[26] J. Cheng, E. Delage, and A. Lisser. Distributionally robust stochastic knapsack problem. *SIAM Journal on Optimization*, 24(3):1485–1506, 2014. doi: 10.1137/130915315. URL https://doi.org/10.1137/130915315.

[27] J. Cheng, M. Houda, and A. Lisser. Chance constrained 0–1 quadratic programs using copulas. *Optimization Letters*, 9(7):1283–1295, 2015. doi: 10.1007/s11590-015-0854-y.

[28] J. Cheng, C. Gicquel, and A. Lisser. Partial sample average approximation method for chance constrained problems. *Optimization Letters*, 13(4):657–672, 2018. doi: 10.1007/s11590-018-1300-8.

[29] M. Chiang and S. Boyd. Geometric programming duals of channel capacity and rate distortion. *IEEE Transactions on Information Theory*, 50(2):245–258, 2004. doi: 10.1109/tit.2003.822581.

[30] W. Chun-Feng, L. San-Yang, and S. Pei-Ping. Global optimization for sum of geometric fractional functions. *Applied Mathematics and Computation*, 216(8):2263–2270, 2010. ISSN 0096-3003. doi: https://doi.org/10.1016/j.amc.2010.03.061. URL https://www.sciencedirect.com/science/article/pii/S0096300310003152.

[31] D. Chunlin and Y. Liu. Sample average approximation method for chance constrained stochastic programming in transportation model of emergency management. *Systems Engineering Procedia*, 5:137–143, 2012. ISSN 2211-3819. doi: https://doi.org/10.1016/j.sepro.2012.04.022. URL https://www.sciencedirect.com/science/article/pii/S2211381912000653. Safety and Emergency Systems Engineering.

[32] D. L. Clark and P. C. Abolmoali. *Gradient-Based Optimization of Time-Dependent Aircraft Subsystems under Uncertainty*. doi: 10.2514/6.2021-3102. URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-3102.

[33] R. J. Clasen. The solution of the chemical equilibrium programming problem with generalized benders decomposition. *Operations Research*, 32(1):70–79, 1984. doi: 10.1287/opre.32.1.70.

[34] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002. doi: 10.1109/4235.985692.

[35] Q. Dai and J. Yang. A distributionally robust chance-constrained approach for modeling demand uncertainty in green port-hinterland transportation network optimization. *Symmetry*, 12(9), 2020. ISSN 2073-8994. doi: 10.3390/sym12091492. URL https://www.mdpi.com/2073-8994/12/9/1492.

[36] J. J. Dinkel, W. H. Elliott, and G. A. Kochenberger. Computational aspects of cutting-plane algorithms for geometric programming problems. *Mathematical Programming*, 13(1):200–220, 1977. doi: 10.1007/bf01584337.

[37] R. J. Duffin. Dual programs and minimum cost. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):119–123, 1962. doi: 10.1137/0110011.

[38] R. J. Duffin. Cost minimization problems treated by geometric means. *Operations Research*, 10(5):668–675, 1962. doi: 10.1287/opre.10.5.668.

[39] R. J. Duffin and E. L. Peterson. Duality theory for geometric programming. *SIAM Journal on Applied Mathematics*, 14(6):1307–1349, 1966. doi: 10.1137/0114105.

[40] J. Dupačová. Stochastic geometric programming with an application. *Kybernetika*, 46(3):374–386, 2010. URL http://eudml.org/doc/196973.

[41] J. G. Ecker. Geometric programming: Methods, computations and applications. *SIAM Review*, 22(3):338–362, 1980. doi: 10.1137/1022058.

[42] C. Edirisinghe and J. Jeong. Meanndash;variance portfolio efficiency under leverage aversion and trading impact. *Journal of Risk and Financial Management*, 15(3), 2022. ISSN 1911-8074. doi: 10.3390/jrfm15030098. URL https://www.mdpi.com/1911-8074/15/3/98.

[43] L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, 1997. doi: 10.1137/S0895479896298130. URL https://doi.org/10.1137/S0895479896298130.

[44] K.-T. Fang, S. Kotz, and K. W. Ng. Spherically and elliptically symmetric distributions. *Symmetric Multivariate and Related Distributions*, page 26–68, 1990. doi: 10.1007/978-1-4899-2937-2_2.

[45] V. A. Faria, A. Rodrigo de Queiroz, and J. F. DeCarolis. Scenario generation and risk-averse stochastic portfolio optimization applied to offshore renewable energy technologies. *Energy*, 270:126946, 2023. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2023.126946. URL https://www.sciencedirect.com/science/article/pii/S0360544223003407.

[46] D. Fedorov and L. Birglen. Geometric optimization of a self-adaptive robotic leg. *Transactions of the Canadian Society for Mechanical Engineering*, 42(1):49–60, March 2018. doi: 10.1139/tcsme-2017-0010. URL https://publications.polymtl.ca/3133/.

[47] C. A. Floudas. *Deterministic global optimization: Theory, methods, and applications*. Kluwer Academic Publishers, 2000.

[48] C. Foias and A. E. Frazho. *Positive Definite Block Matrices*, pages 547–586. Birkhäuser Basel, Basel, 1990. ISBN 978-3-0348-7712-1. doi: 10.1007/978-3-0348-7712-1_16. URL https://doi.org/10.1007/978-3-0348-7712-1_16.

[49] R. J. Fonseca, W. Wiesemann, and B. Rustem. Robust international portfolio management. *Comput. Manag. Sci.*, 9(1):31–62, 2012. doi: 10.1007/s10287-011-0132-0. URL https://doi.org/10.1007/s10287-011-0132-0.

[50] C. Frank. An algorithm for geometric programming. *Recent Advances in Optimization Techniques*, (3513):145–162, 1966.

[51] A. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 10 1972. doi: 10.1007/BF00934810.

[52] S. Ghosal and W. Wiesemann. The distributionally robust chance-constrained vehicle routing problem. *Operations Research*, 68(3):716–732, 2020. doi: 10.1287/opre.2019.1924. URL https://doi.org/10.1287/opre.2019.1924.

[53] W. Gochet and Y. Smeers. A branch-and-bound method for reversed geometric programming. *Operations Research*, 27(5):982–996, 1979. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/170062.

[54] K.-C. Goh, L. Turan, M. G. Safonov, G. P. Papavassilopoulos, and J. Ly. Biaffine matrix inequality properties and computational methods. *Proceedings of 1994 American Control Conference - ACC '94*, 1:850–855 vol.1, 1994.

[55] B. Gopalakrishnan, A. K. Singh, K. M. Krishna, and D. Manocha. Solving chance-constrained optimization under nonparametric uncertainty through hilbert space embedding. *IEEE Transactions on Control Systems Technology*, 30(3):901–916, 2022. doi: 10.1109/TCST.2021.3091315.

[56] J. Gorski, F. Pfeuffer, and K. Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66 (3):373–407, December 2007. doi: 10.1007/s00186-007-0161-1. URL https://ideas.repec.org/a/spr/mathme/v66y2007i3p373-407.html.

[57] P. F. Gorski Jochen and K. Kathrin. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, page 373–467, 2007. ISSN 1432-5217. doi: 10.1007/s00186-007-0161-1.

[58] B. Guan. *Optimal Resource Allocation for video streaming over cognitive radio network via Geometric Programming*, 2021. doi: 10.32920/ryerson.14653935.v1.

[59] M. Haleem. On the capacity and transmission techniques of massive mimo systems. *Wireless Communications and Mobile Computing*, 2018:1–9, 07 2018. doi: 10.1155/2018/9363515.

[60] G. Hanasusanto, D. Kuhn, S. Wallace, and S. Zymler. Distributionally robust multi-item newsvendor problems with multimodal demand distributions. *Mathematical Programming*, 152, 04 2014. doi: 10.1007/s10107-014-0776-y.

[61] L. J. Hong, Y. Yang, and L. Zhang. Sequential convex approximations to joint chance constrained programs: A monte carlo approach. *Operations Research*, 59(3):617–630, 2011. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/23013131.

[62] J. Hopfield and D. Tank. Neural computation of decisions in optimization problems. *Biological cybernetics*, 52:141–52, 02 1985. doi: 10.1007/BF00339943.

[63] A. Hosseini, J. Wang, and S. M. Hosseini. A recurrent neural network for solving a class of generalized convex optimization problems. *Neural Networks*, 44, 2013. ISSN 08936080. doi: 10.1016/j.neunet.2013.03.010.

[64] M. Houda and A. Lisser. On the use of copulas in joint chance-constrained programming. In *Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems*, ICORES 2014, page 72–79, Setubal, PRT, 2014. SCITEPRESS - Science and Technology Publications, Lda. ISBN 9789897580178. doi: 10.5220/0004831500720079. URL https://doi.org/10.5220/0004831500720079.

[65] R. Jagannathan. A stochastic geometric programming problem with multiplicative recourse. *Operations Research Letters*, 9(2):99–104, 1990. ISSN 0167-6377. doi: https://doi.org/10.1016/0167-6377(90)90048-A. URL https://www.sciencedirect.com/science/article/pii/016763779090048A.

[66] A. Janushevskis, J. Auzins, A. Melnikovs, and A. Gerina-Ancane. *Shape Optimization of Mechanical Components for Measurement Systems*. 03 2012. ISBN 978-953-51-0128-4. doi: 10.5772/36297.

[67] N. K. Jha. A Discrete Data Base Multiple Objective Optimization of Milling Operation Through Geometric Programming. *Journal of Engineering for Industry*, 112(4):368–374, 11 1990. ISSN 0022-0817. doi: 10.1115/1.2899601. URL https://doi.org/10.1115/1.2899601.

[68] M. Jiang, Z. Meng, and R. Shen. Partial exactness for the penalty function of biconvex programming. *Entropy*, 23(2):132, Jan 2021. ISSN 1099-4300. doi: 10.3390/e23020132. URL http://dx.doi.org/10.3390/e23020132.

[69] M. Jiang, Z. Meng, and R. Shen. Partial exactness for the penalty function of biconvex programming. *Entropy*, 23(2), 2021. ISSN 1099-4300. doi: 10.3390/e23020132.

[70] M. N. Jouini and R. T. Clemen. Copula models for aggregating expert opinions. *Operations Research*, 44(3):444–457, 1996. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/171704.

[71] Y. Kabanov and C. Klppelberg. A geometric approach to portfolio optimization in models with transaction costs. *Finance and Stochastics*, 8(2):207–227, 2004. doi: 10.1007/s00780-003-0114-3.

[72] M. Kennedy and L. Chua. Neural networks for nonlinear programming. *IEEE Transactions on Circuits and Systems*, 35(5):554–562, 1988. doi: 10.1109/31.1783.

[73] A. Kerboua and R. Kelaiaia. Recurrent neural network optimization for wind turbine condition prognosis. *Diagnostyka*, 23, 2022. ISSN 24495220. doi: 10.29354/diag/151608.

[74] R. Khanjani, M. Tavana, H. Fukuyama, and D. Di Caprio. Fuzzy chance-constrained geometric programming: The possibility, necessity and credibility approaches. *Operational Research*, 17:67–97, 04 2017. doi: 10.1007/s12351-015-0216-7.

[75] R. Khanjani Shiraz, S. Khodayifar, and P. Pardalos. Copula theory approach to stochastic geometric programming. *Journal of Global Optimization*, 81:1–34, 10 2021. doi: 10.1007/s10898-021-01062-7.

[76] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002. doi: 10.1137/s1052623499363220.

[77] R. K.Miller and A. N. Michel. *Ordinary Differential Equations*. Academic Press, 1981.

[78] K. Kortanek, X. Xu, and Y. Ye. An infeasible interior-point algorithm for solving primal and dual geometric programs. *Math. Program.*, 76:155–181, 01 1996. doi: 10.1007/BF02614382.

[79] K. Kortanek, X. Xu, and Y. Ye. An infeasible interior-point algorithm for solving primal and dual geometric programs. *Math. Program.*, 76:155–181, 01 1996. doi: 10.1007/BF02614382.

[80] A. Kulkarni. Modified probability collectives approach for solving engineering problems. 06 2013. doi: 10.13140/RG.2.2.33706.82881.

[81] E. Köker and A. B. Altan-Sakarya. Chance constrained optimization of booster chlorination in water distribution networks. *CLEAN - Soil, Air, Water*, 43(5):717–723, 2014. doi: 10.1002/clen.201400119.

[82] D. Lee, C. Han, S. Kang, and G. JANG. *Chance-constrained optimization for active distribution networks with virtual power lines*, 2023. doi: 10.2139/ssrn.4353967.

[83] D. Li and X. Dai. Power control in cooperative cognitive radio networks by geometric programming. *2009 15th Asia-Pacific Conference on Communications*, 2009. doi: 10.1109/apcc.2009.5375674.

[84] N. Li, I. Kolmanovsky, and A. Girard. An analytical safe approximation to joint chance-constrained programming with additive gaussian noises. *IEEE Transactions on Automatic Control*, 66(11):5490–5497, 2021. doi: 10.1109/TAC.2021.3051000.

[85] M.-H. Lin and J.-F. Tsai. Finding multiple optimal solutions of signomial discrete programming problems with free variables. *Optimization and Engineering*, 12(3):425–443, 2011. doi: 10.1007/s11081-011-9137-3.

[86] S. H. Ling, H. H. C. Iu, K. Y. Chan, H. K. Lam, B. C. W. Yeung, and F. H. Leung. Hybrid particle swarm optimization with wavelet mutation and its industrial applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(3):743–763, 2008. doi: 10.1109/TSMCB.2008.921005.

[87] A. Lisser, J. Liu, and S. Peng. Rectangular chance constrained geometric optimization. *Optimization and Engineering*, 21:1573–2924, 2020. doi: https://doi.org/10.1007/s11081-019-09460-3.

[88] J. Liu, A. Lisser, and Z. Chen. Stochastic geometric optimization with joint probabilistic constraints. *Operations Research Letters*, 44(5):687–691, 2016. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2016.08.002. URL https://www.sciencedirect.com/science/article/pii/S0167637716300761.

[89] J. Liu, A. Lisser, and Z. Chen. Distributionally robust chance constrained geometric optimization. *Mathematics of Operations Research*, 47(4):2950–2988, 2022. doi: 10.1287/moor.2021.1233. URL https://doi.org/10.1287/moor.2021.1233.

[90] Q. Liu, Z. Guo, and J. Wang. A one-layer recurrent neural network for constrained pseudoconvex optimization and its application for dynamic portfolio optimization. *Neural Networks*, 26, 2012. ISSN 08936080. doi: 10.1016/j.neunet.2011.09.001.

[91] Q. Liu, S. Yang, and J. Wang. A collective neurodynamic approach to distributed constrained optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 28(8):1747–1758, 2017. doi: 10.1109/TNNLS.2016.2549566.

[92] S.-T. Liu. Posynomial geometric programming with parametric uncertainty. *European Journal of Operational Research*, 168(2):345–353, January 2006. URL https://ideas.repec.org/a/eee/ejores/v168y2006i2p345-353.html.

[93] S.-T. Liu. Geometric programming with fuzzy parameters in engineering optimization. *International Journal of Approximate Reasoning*, 46:484–498, 12 2007. doi: 10.1016/j.ijar.2007.01.004.

[94] X. Liu and M. Zhou. A one-layer recurrent neural network for non-smooth convex optimization subject to linear inequality constraints. *Chaos, Solitons and Fractals*, 87, 2016. ISSN 09600779. doi: 10.1016/j.chaos.2016.03.009.

[95] W. Lodwick, F. Newman, and A. Neumaier. Optimization under uncertainty: Methods and applications in radiation therapy. *10th IEEE International Conference on Fuzzy Systems. (Cat. No.01CH37297)*. doi: 10.1109/fuzz.2001.1008877.

[96] J. Luedtke. A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support. *Mathematical Programming*, 146(1–2):219–244, 2013. doi: 10.1007/s10107-013-0684-6.

[97] J. Luedtke and S. Ahmed. A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, 19(2):674–699, 2008. doi: 10.1137/070702928. URL https://doi.org/10.1137/070702928.

[98] E. A. Melissaratos and D. L. Souvaine. On solving geometric optimization problems using shortest paths. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*, SCG '90, page 350–359, New York, NY, USA, 1990. Association for Computing Machinery. ISBN 0897913620. doi: 10.1145/98524.98600. URL https://doi.org/10.1145/98524.98600.

[99] B. L. Miller and H. M. Wagner. Chance constrained programming with joint constraints. *Operations Research*, 13:930–945, 1965.

[100] H. Nasseri and Z. Alizadeh. Optimized solution of a two-bar truss nonlinear problem using fuzzy geometric programming. *Journal of Nonlinear Analysis and Application*, 2014:1–9, 01 2014. doi: 10.5899/2014/jnaa-00230.

[101] A. Nazemi. A dynamic system model for solving convex nonlinear optimization problems. *Communications in Nonlinear Science and Numerical Simulation*, 17(4):1696–1705, 2012. ISSN 1007-5704. doi: https://doi.org/10.1016/j.cnsns.2011.08.035. URL https://www.sciencedirect.com/science/article/pii/S1007570411004709.

[102] A. Nemirovski and A. Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2007. doi: 10.1137/050622328. URL https://doi.org/10.1137/050622328.

[103] A. Ojha and R. R. Ota. Multi-objective geometric programming problem with karushkuhntucker condition using-constraint method. *RAIRO - Operations Research*, 48(4):429–453, 2014. doi: 10.1051/ro/2014016.

[104] C. Ordoudis, V. A. Nguyen, D. Kuhn, and P. Pinson. Energy and reserve dispatch with distributionally robust joint chance constraints. *Operations Research Letters*, 49(3): 291–299, 2021. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2021.01.012. URL https://www.sciencedirect.com/science/article/pii/S0167637721000213.

[105] A. Patton. Copula-based models for financial time series. *Oxford Financial Research Centre, OFRC Working Papers Series*, 01 2008. doi: 10.1007/978-3-540-71297-8_34.

[106] S. Peng, A. Lisser, V. V. Singh, N. Gupta, and E. Balachandar. Games with distributionally robust joint chance constraints. *Optim. Lett.*, 15(6):1931–1953, 2021. doi: 10.1007/s11590-021-01700-9. URL https://doi.org/10.1007/s11590-021-01700-9.

[107] D. D. Perlumutter. Geometric programming, duffin, r. j., peterson, e. l., and zener, c., john wiley, new york(1967). 278 pages. *AIChE Journal*, 13(4):829–830, 1967. doi: 10.1002/aic.690130408.

[108] K. Postek, D. den Hertog, and B. Melenberg. Computationally tractable counterparts of distributionally robust constraints on risk measures. *SIAM Review*, 58(4):603–650, 2016. doi: 10.1137/151005221. URL https://doi.org/10.1137/151005221.

[109] A. Prekopa. *ON PROBABILISTIC CONSTRAINED PROGRAMMING*, pages 113–138. Princeton University Press, Princeton, 1971. ISBN 978-1-4008-6993-0. doi: doi:10.1515/9781400869930-009. URL https://doi.org/10.1515/9781400869930-009.

[110] A. Prékopa. A class of stochastic programming decision problems. *Optimization*, 3: 349–354, 1972.

[111] S. Qin and X. Xue. A two-layer recurrent neural network for nonsmooth convex optimization problems. *IEEE Transactions on Neural Networks and Learning Systems*, 26, 2015. ISSN 21622388. doi: 10.1109/TNNLS.2014.2334364.

[112] S. Qin, Y. Liu, X. Xue, and F. Wang. A neurodynamic approach to convex optimization problems with general constraint. *Neural Networks*, 84, 2016. ISSN 18792782. doi: 10.1016/j.neunet.2016.08.014.

[113] S.-J. Qu, K.-C. Zhang, and Y. Ji. A new global optimization algorithm for signomial geometric programming via lagrangian relaxation. *Applied Mathematics and Computation*, 184(2):886–894, 2007. ISSN 0096-3003. doi: https://doi.org/10.1016/j.amc.2006.05.208. URL https://www.sciencedirect.com/science/article/pii/S0096300306007776.

[114] R. Reiszig. J. lasalle and s. lefschetz, stability by liapunov's direct method with applications. vii + 134 s. new york/london 1961. academic press. preis geb. $ 5.50. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 42(10-11):514–514, 1962. doi: https://doi.org/10.1002/zamm.

19620421022. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19620421022.

[115] M. J. Rijckaert and X. M. Martens. A condensation method for generalized geometric programming. *Mathematical Programming*, 11(1):89–93, 1976. doi: 10.1007/bf01580373.

[116] L. A. Roald, D. Pozo, A. Papavasiliou, D. K. Molzahn, J. Kazempour, and A. Conejo. Power systems optimization under uncertainty: A review of methods and applications. *Electric Power Systems Research*, 214:108725, 2023. doi: 10.1016/j.epsr.2022.108725.

[117] R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis*. Springer, Berlin, Heidelberg, 1998. ISBN 978-3-642-02431-3. doi: 10.1007/978-3-642-02431-3.

[118] E. Rosenberg. On solving a primal geometric program by partial dual optimization. *Mathematical Programming*, 21(1):319–330, 1981. doi: 10.1007/bf01584252.

[119] M. Saraj. Solving a posynomial geometric programming problem with fully fuzzy approach. *Yugoslav Journal of Operations Research*, 29(2):203–220, 2019. ISSN 2334-6043.

[120] J. Sassen, B. Heeren, K. Hildebrandt, and M. Rumpf. Geometric optimization using nonlinear rotation-invariant coordinates. *Computer Aided Geometric Design*, 77:101829, 2020. ISSN 0167-8396. doi: https://doi.org/10.1016/j.cagd.2020.101829. URL https://www.sciencedirect.com/science/article/pii/S0167839620300169.

[121] C. Shang and F. You. Distributionally robust optimization for planning and scheduling under uncertainty. *Computers Chemical Engineering*, 110:53–68, 2018. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2017.12.002. URL https://www.sciencedirect.com/science/article/pii/S009813541730426X.

[122] P. Shen and H. Jiao. A new rectangle branch-and-pruning approach for generalized geometric programming. *Applied Mathematics and Computation*, 183(2):1027–1038, 2006. ISSN 0096-3003. doi: https://doi.org/10.1016/j.amc.2006.05.137. URL https://www.sciencedirect.com/science/article/pii/S0096300306006382.

[123] P. Shen and K. Zhang. Global optimization of signomial geometric programming using linear relaxation. *Applied Mathematics and Computation*, 150(1):99–114, 2004. ISSN 0096-3003. doi: https://doi.org/10.1016/S0096-3003(03)00200-5. URL https://www.sciencedirect.com/science/article/pii/S0096300303002005.

[124] Y. Shen, E. Y. Lam, and N. Wong. A signomial programming approach for binary image restoration by penalized least squares. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(1):41–45, 2008. doi: 10.1109/TCSII.2007.907751.

[125] B. Singh and J.-P. Watson. *Chance-constrained optimization for critical infrastructure protection.*, 2018. doi: 10.2172/1474266.

[126] A. Sklar. Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 8:229–231, 1959.

[127] J.-J. Slotine. *Applied nonlinear control*. Prentice Hall, Englewood Cliffs (N.J.), right 1991. ISBN 0-13-040890-5.

[128] J.-J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, 1991. ISBN 9780130408907. URL https://books.google.fr/books?id=cwpRAAAAMAAJ.

[129] A. Stupar, J. A. Taylor, and A. Prodic. Posynomial models of inductors for optimization of power electronic systems by geometric programming. *2016 IEEE 17th Workshop on Control and Modeling for Power Electronics (COMPEL)*, 2016. doi: 10.1109/compel.2016.7556660.

[130] Y. Sun, G. Aw, R. Loxton, and K. L. Teo. Chance-constrained optimization for pension fund portfolios in the presence of default risk. *European Journal of Operational Research*, 256, 2017. ISSN 03772217. doi: 10.1016/j.ejor.2016.06.019.

[131] S. Tassouli and A. Lisser. Solving linear programs with joint probabilistic constraints with dependent rows using a dynamical neural network. *Results in Control and Optimization*, 9:100178, 2022. ISSN 2666-7207. doi: https://doi.org/10.1016/j.rico.2022.100178. URL https://www.sciencedirect.com/science/article/pii/S2666720722000509.

[132] S. Tassouli and A. Lisser. A neural network approach to solve geometric programs with joint probabilistic constraints. *Mathematics and Computers in Simulation*, 205:765–777, 2023. ISSN 0378-4754. doi: https://doi.org/10.1016/j.matcom.2022.10.025. URL https://www.sciencedirect.com/science/article/pii/S0378475422004384.

[133] R. Toscano and P. Lyonnet. Evolutionary algorithms for solving quasi geometric programming problems. In *Proceedings of the International Conference on Evolutionary Computation - Volume 1: ICEC, (IJCCI 2010)*, pages 163–169. INSTICC, SciTePress, 2010. ISBN 978-989-8425-31-7. doi: 10.5220/0003071901630169.

[134] J.-F. Tsai. Global optimization for signomial discrete programming problems in engineering design. *Engineering Optimization*, 42(9):833–843, 2010. doi: 10.1080/03052150903456485.

[135] M.-L. Tseng, K.-J. Wu, J. Hu, and C.-H. Wang. Decision-making model for sustainable supply chain finance under uncertainties. *International Journal of Production Economics*, 205:30–36, 2018. doi: 10.1016/j.ijpe.2018.08.024.

[136] M. E. G. Urias, E. N. Sanchez, and L. J. Ricalde. Electrical microgrid optimization via a new recurrent neural network. *IEEE Systems Journal*, 9, 2015. ISSN 19379234. doi: 10.1109/JSYST.2014.2305494.

[137] S. Uryasev and P. Pardalos. *Stochastic Optimization: Algorithms and Applications*. Applied Optimization. Springer US, 2013. ISBN 9781475765946. URL https://books.google.fr/books?id=B_fiBwAAQBAJ.

[138] V. Visweswaran and C. Floudast. A global optimization algorithm (gop) for certain classes of nonconvex nlps—ii. application of theory and test problems. *Computers Chemical Engineering*, 14(12):1419–1434, 1990. ISSN 0098-1354. doi: https://doi.org/10.1016/0098-1354(90)80021-3. URL https://www.sciencedirect.com/science/article/pii/0098135490800213.

[139] T. W. Wall, D. Greening, and R. E. Woolsey. Or practice—solving complex chemical equilibria using a geometric-programming based technique. *Operations Research*, 34(3):345–355, 1986. doi: 10.1287/opre.34.3.345.

[140] S. Wang, L. Pang, H. Guo, and H. Zhang. Distributionally robust optimization with multivariate second-order stochastic dominance constraints with applications in portfolio optimization. *Optimization*, 0(0):1–24, 2022. doi: 10.1080/02331934.2022.2048382. URL https://doi.org/10.1080/02331934.2022.2048382.

[141] Y. Xia and J. Wang. A recurrent neural network for solving linear projection equations. *Neural Netw.*, 13(3):337–350, apr 2000. ISSN 0893-6080. doi: 10.1016/S0893-6080(00)00019-8. URL https://doi.org/10.1016/S0893-6080(00)00019-8.

[142] Y. Xia and J. Wang. A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(7):1385–1394, 2004. doi: 10.1109/TCSI.2004.830694.

[143] W. Xie and S. Ahmed. On deterministic reformulations of distributionally robust joint chance constrained optimization problems. *SIAM Journal on Optimization*, 28(2):1151–1182, 2018. doi: 10.1137/16M1094725. URL https://doi.org/10.1137/16M1094725.

[144] J. Yan, H. Kong, and Z. Man. Recurrent neural network-based nonlinear optimization for braking control of electric vehicles. *Energies*, 15, 2022. ISSN 19961073. doi: 10.3390/en15249486.

[145] A. Zare, C. Y. Chung, J. Zhan, and S. O. Faried. A distributionally robust chance-constrained milp model for multistage distribution system planning with uncertain renewables and loads. *IEEE Transactions on Power Systems*, 33(5):5248–5262, 2018. doi: 10.1109/TPWRS.2018.2792938.

[146] C. Zener. A mathematical aid in optimizing engineering designs. *Proceedings of the National Academy of Sciences*, 47(4):537–539, 1961. doi: 10.1073/pnas.47.4.537.

[147] C. Zener. A further mathematical aid in optimizing engineering designs. *Science*, 136 (3513):330–330, 1962. doi: 10.1126/science.136.3513.330-c.

[148] Y. Zhang, S. Shen, and S. A. Erdogan. Distributionally robust appointment scheduling with moment-based ambiguity set. *Operations Research Letters*, 45(2):139–144, 2017. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2017.01.010. URL https://www.sciencedirect.com/science/article/pii/S0167637717300688.