A neurodynamic algorithm for dependent joint chance constrained geometric programs

Siham Tassouli, Abdel Lisser

Results in Control and Optimization 12 (2023) 100275

Contents lists available at ScienceDirect

Results in Control and Optimization

journal homepage: www.elsevier.com/locate/rico

A neurodynamic algorithm for dependent joint chance constrained geometric programs

Siham Tassouli *, Abdel Lisser

Université Paris Saclay, CNRS, CentraleSupélec, Laboratoire des Signaux et Systèmes (L2S), 3, rue Joliot Curie, 91192 Gif sur Yvette Cedex, France

ARTICLE INFO

MSC:
00-01
99-00

Keywords:
Copula theory
Biconvex optimization
Dynamical neural network
Partial KKT system
Chance constrained geometric programs

ABSTRACT

This research studies the use of copula theory to model dependencies in joint probabilistic constrained geometric programs with dependent rows. The row vectors are assumed to follow an elliptical distribution, and their dependencies are modeled through a Gumbel–Hougaard copula. We use a log transformation to convert the chance-constrained geometric program into a deterministic optimization problem. Then we solve the resulting deterministic program using a dynamical neural network. The stability and convergence of the proposed neural network approach are demonstrated. The primary characteristic of our framework is its ability to solve the dependent joint chance constrained geometric programs without resorting to any convex approximation methods. This feature sets our approach apart from the current state-of-the-art solving techniques. The neurodynamic algorithm is finally applied to solve three geometric optimization problems.

1. Introduction

Chance-constrained optimization appears as an approach for solving optimization problems where uncertainty is present in the constraints. Chance-constrained optimization dates back to the early work of Charnes & Cooper [1]. Since then, chance-constrained optimization attracted the attention of researchers and was applied to deal with numerous real-world problems, e.g., power management [2], finance [3,4], scheduling [5], healthcare [6] and wireless communication [7]. Joint chance-constrained optimization is a special case of chance-constrained optimization that has the advantage of assuring the satisfaction of all constraints at a certain level. Lejeune et al. [8] propose a Boolean modeling framework to solve linear programs with joint probabilistic constraints. Nemirovski et al. [9] give some convex approximations to solve joint chance-constrained optimization problems. Chen et al. [10] provide worst-case bounds to approximate the solution for both individual and joint chance-constrained problems. Cheng & Lisser [11] use a piecewise linear approximation and a piecewise tangent approximation to come up with an upper and a lower bounds for linear programs with joint probabilistic constraints. Based on the optimality conditions of the deterministic equivalent of the linear programs with joint probabilistic constraints, Tassouli & Lisser [12] design a convergent neurodynamic neural network to approximate the optimal value. An other way to deal with uncertainties in chance-constrained optimization is the fuzzy theory initialized by Zadeh [13] in the early 1960s. Since then, the theory has been applied to solve a variety of problems. Sharma et al. [14] introduce a novel approach to address the challenges of solving multi-objective bi-level chance-constrained optimization problems within an intuitionistic fuzzy framework. The technological coefficients and objective function coefficients are represented using Triangular Intuitionistic fuzzy numbers, while normal random variables are employed to represent resource coefficients. Rani et al. [15] present some generalized techniques for solving intuitionistic fuzzy multi-objective non-linear optimization problems. A detailed survey on fuzzy chance-constrained can be found in [16].

* Corresponding author.
E-mail addresses: siham.tassouli@centralesupelec.fr (S. Tassouli), abdel.lisser@centralesupelec.fr (A. Lisser).

https://doi.org/10.1016/j.rico.2023.100275
Received 22 May 2023; Received in revised form 3 July 2023; Accepted 22 July 2023
Available online 26 July 2023

Geometric programming (GP) is a mathematical technique designed to determine the constrained minimum value of a generalized polynomial objective function. The first work involving geometric optimization appeared in 1971 by Kochenberger [17]. Geometric optimization has been a subject of study for several decades. Maranas & Fouldas [18] propose a deterministic global optimization algorithm that can locate the global minimum of generalized geometric problems. Rijckaert and Martens [19] propose a dual condensation algorithm to come up with a solution to generalized geometric programs. Bricker and Rajgopal [20] describe an algorithm for the geometric programming dual problem using an adaptation of the generalized LP algorithm. Kortanek et al. [21] present an infeasible interior-point algorithm for solving primal and dual geometric programs. The application of geometric programming spans various domains, such as engineering design, power control, and finance. However, the conventional deterministic geometric programming method relies on deterministic values for decision variable coefficients and exponents. This approach becomes inadequate when dealing with complex real-life problems that involve uncertainty and impreciseness in parameters and data. Mondal et al. [22] solve chance-constrained geometric programming problems by using triangular and trapezoidal uncertainty distributions for the uncertain variables. Liu [23] proposes a procedure to obtain the lower and upper bounds of the objective function for posynomial geometric programming problems in the presence of uncertain cost and constraint parameters and [24] develops a procedure to determine the fuzzy objective value of fuzzy posynomial geometric programming problems, where the exponents of decision variables in the objective function, the cost and constraint coefficients, and the right-hand sides are represented by fuzzy numbers. Shiraz & Fukuyama [25] introduce a rough geometric programming method that combines deterministic geometric programming with rough set theory. The proposed method exhibits three main characteristics: the coefficients in the objective function and constraints are treated as rough variables, the expected-value operator is applied to rough variables, and it can determine both the lower and upper bounds of the objective function at a specific trust level. Shiraz et al. [26] employ fuzzy variables to formalize the uncertainty surrounding the coefficients of GP problems. Three variants of chance-constrained GP are formulated based on the possibility, necessity, and credibility approaches. The paper demonstrates how these variants can be transformed into equivalent deterministic GP problems, which can be solved using the duality algorithm. Nodeh and al. [27] explore the constrained shortest path problem, specifically focusing on the scenario where the arc resources follow a dependent normal distribution. A model is proposed to maximize the probability of meeting all constraints while staying within a specified limit. A Copula-based marginal distribution approach is employed to handle the dependency between the constraint matrix rows, utilizing an appropriate Archimedean Copula. The joint chance-constrained problems are then transformed into deterministic problems using second-order cone programming.

The neurodynamic system approach is a significant method for addressing optimization problems. Artificial recurrent neural networks are utilized as a tool to convert optimization problems into a specific dynamic system represented by first-order differential equations. This dynamic system is expected to converge to a static state or equilibrium point, which corresponds to the solution of the original optimization problem, starting from an initial point. Additionally, neural networks designed for optimization problems can be implemented in hardware using integrated circuits, making them readily deployable. Two compelling features of neural networks for optimization problems are parallel information processing and hardware implementability. Neural networks have inherent parallel processing capabilities. The structure of neural networks allows for the simultaneous evaluation of multiple inputs and the computation of corresponding outputs. This parallelism enables efficient and concurrent processing of information, resulting in faster optimization performance compared to sequential algorithms. Neural networks can be implemented using specialized hardware, such as 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 leveraging hardware resources, neural networks can be deployed in real-time applications or embedded systems, enabling efficient and rapid optimization in various domains. Over the past few decades, recurrent neural networks (RNNs) have been extensively studied for solving optimization problems. One of the early breakthroughs in this field was made by Hopfield & Tank in 1985 [28], where they developed a linear programming neural network. This network was specifically designed for online optimization applications. Since then, numerous RNN architectures have been proposed for solving constrained optimization problems. Xia & Feng [29] introduce an RNN for solving nonlinear projection equations. The neural network architecture comprises a single layer and is well-suited for parallel implementation, leveraging the advantages of parallel computing. Under mild conditions of the underlying nonlinear mapping, the neural network is proven to converge to an accurate solution. Liu & Qin [30] present a neurodynamic approach for addressing nonlinear optimization problems that involve affine equality and convex inequality constraints. By incorporating the time-varying auxiliary function, the proposed neural network is designed to converge to the feasible region of the optimization problem in a limited CPU time. Once the neural network enters the feasible region, it remains within it throughout the optimization process. Leung & Wang [31] introduce a collaborative neurodynamic approach for tackling multiobjective optimization problems. The approach aims to achieve two key objectives: Pareto optimality and solution diversity. To address the multiple objectives, a weighted Chebyshev function is employed to scalarize the objectives. This scalarization enables the transformation of the multiobjective problem into a single-objective problem. Leung and al. [32] apply neurodynamic optimization techniques to portfolio selection, considering both variable weights and cardinality constraints. Neurodynamic approaches are also applied to solve chance-constrained optimization. Nazemi & Tahmasbi [33] present a neural network model to solve linear optimization problems with individual chance constraints. Tassouli & Lisser [34] propose a neural network approach to solve geometric programs with joint probabilistic constraints with normally distributed coefficients and independent matrix row vectors. Tassouli & Lisser [12] solve linear programs with joint probabilistic constraints with normally distributed and dependent rows using a dynamical neural network.

Building upon prior research, this paper introduces a novel neurodynamic approach for solving dependent joint chance-constrained geometric programs where the row vectors are assumed to follow an elliptical distribution and their dependencies are modeled through a Gumbel–Hougaard copula. The main contributions of the presented network in this paper can be summarized as follows.

(i) The paper addresses the challenging problem of solving dependent joint chance-constrained geometric programs. These programs involve interdependent variables and constraints, adding complexity to the optimization process. The proposed network offers a solution approach tailored to this problem setting.
(ii) The presented network incorporates a Gumbel–Hougaard copula to model the dependencies among the row vectors. This copula allows for capturing the underlying dependencies accurately, enabling more realistic and precise optimization solutions.
(iii) To the best of our knowledge, the only work involving dependent joint chance constrained geometric program is the one of Shiraz et al. [35]. Compared with [35], the presented neurodynamic optimization model in this paper does not rely on any convex or linear approximation to solve the optimization problem.
(iv) We establish the neurodynamic model's stability and global convergence properties using Lyapunov theory and applying LaSalle's invariance principle.
(v) Our numerical experiments show that our dynamical neural network provides a high-quality upper bounds for the original geometric program when compared to the state-of-the-art and assess the robustness of the solutions by generating random samples of the variables and checking the satisfaction of the constraints.

The rest of the paper is organized as follows. In Section 2, we introduce the copulas theory as a tool to deal with dependencies between the random variables. In Section 3, we derive the deterministic equivalent of the studied problem and present its optimality conditions. In Section 4, we propose a dynamical neural network to approximate the optimal value and show its stability and convergence. In Section 5, we apply our proposed algorithm to solve three geometric programs.

2. Preliminaries

To capture the dependencies between the random coefficients, we use the copula theory to derive joint distributions. Fisher gave two mean reasons why copulas are of interest to statisticians: First, they provide scale-free measures of statistical dependence; and second, they enable building classes of joint distributions. The Copula theory dates back to 1959 [36]. Since 1990s, the interest in the copula theory grew markedly. Jouini & Clemen [37] discuss the use of multivariate distributions that are functions of their marginals for aggregating information from various sources. Patton [38] presents a review of the many applications of copulas in finance and economics. Houda & Lisser [39] use copulas to model the dependence between the random variables in joint chance-constrained optimization. Liu et al. [40] present a collective neurodynamic approach with multiple interconnected recurrent neural networks for distributed constrained optimization.

This section serves as an introduction to the concept of a copula and the associated properties that are used in this paper.

Definition 1. A copula, denoted as $C : [0,1]^d \longrightarrow [0,1]$, is a joint cumulative distribution function with a dimension of d. It is characterized by having uniformly distributed marginals within the interval $[0,1]$.

Theorem 1 (*Sklar's Theorem 1959*)**.** *Let F be a d-dimensional cumulative distribution function with marginals $F_1, F_2, \dots, F_d$. In this case, there exists a copula C such that*

$$\forall x \in \mathbb{R}^d, \quad F(x) = C(F_1(x_1), F_2(x_2), \dots, F_d(x_d)).$$

If all the marginals $F_1, F_2, \dots, F_d$ are continuous, then the copula C is unique and given by

$$C(u) = F({F_1}^{-1}(u_1), {F_2}^{-1}(u_2), \dots, {F_d}^{-1}(u_d)),$$

Otherwise, C is uniquely determined within the range $rang(F_1) \times rang(F_2) \times \cdots \times rang(F_d)$.

Definition 2. If a copula C has a density, then it is expressed as follows

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

To capture dependencies between stochastic parameters, explicit copulas known as Archimedean copulas are commonly employed, as many copulas, such as the Gaussian copula, lack an explicit analytical expression.

Definition 3. A copula C is Archimedean if it is represented as follows

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

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 $\psi^{[-1]}(t; \theta) = \begin{cases} \psi^{-1}(t; \theta) \text{ if } 0 \le t \le \psi(0; \theta) \\ 0 \text{ if } t \ge \psi(0; \theta) \end{cases}$ and θ is a dependency parameter.

We give in Table 1 some examples of Archimedean copulas often used in the literature.

In this research, we adopt the assumption that the random vectors follow an elliptical distribution. Consequently, we provide a brief overview of the definitions and properties associated with the elliptical family.

Table 1
Examples of some commonly used Archimedean copulas.

Name of copula	Parameter θ	Generator $\psi_\theta(t)$
Clayton	$\theta > 0$	$\frac{1}{\theta}(t^{-\theta} - 1)$
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 2
Table of selected elliptical distributions.

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	$\frac{2\pi\sqrt{t}}{e^{\pi\sqrt{t}}-e^{-\pi\sqrt{t}}}$	$\frac{e^{-t^2}}{(1+e^{-t^2})^2}$	$\ln(\frac{\alpha}{1-\alpha})$	$\frac{1}{2} \leq \alpha \leq 1$

Definition 4. An n-dimensional vector X follows an elliptical distribution if there exists a function Ψ such that its characteristic function is given by

$$\phi(z) = \mathbb{E}[e^{iz^T X}] = e^{iz^T \mu}\Psi(z^T \Sigma z). \tag{1}$$

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

$$f(x) = \frac{C}{\sqrt{\det\Sigma}} g\left(\sqrt{(x-\mu)^T \Sigma^{-1}(x-\mu)}\right). \tag{2}$$

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 1. A collection of various multivariate elliptical distributions, their characteristic generators, radial densities, quantile functions of the standard distribution, and the convexity conditions of the quantile function are presented in Table 2.

3. Chance-constrained geometric programming

A standard geometric program is given as follows

$$\begin{aligned} &\min_{t\in\mathbb{R}^M_{++}} \sum_{i=1}^{I_0} c_i \prod_{j=1}^{M} t_j^{a_{ij}}, \\ &\text{s.t} \quad \sum_{i=1}^{I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \leq 1, k = 1, \dots, K, \end{aligned} \tag{3}$$

where $c_i > 0$, $i = 1, \dots, I_k$, $k = 0, 1, \dots, K$ and a_{ij}, $i = 1, \dots, I_k$, $j = 1, \dots, M$ are real constants.

The joint chance-constrained version of the geometric program is then given by

$$\begin{aligned} &\min_{t\in\mathbb{R}^M_{++}} \mathbb{E}\left[\sum_{i=1}^{I_0} c_i \prod_{j=1}^{M} t_j^{a_{ij}}\right], \\ &\text{s.t} \quad \mathbb{P}\left(\sum_{i=1}^{I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \leq 1, k = 1, \dots, K\right) \geq \alpha, \end{aligned} \tag{4}$$

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

Assumption 1. We assume that the row vectors are dependent and elliptically distributed and that the dependence is captured by a Gumbel–Hougaard copula, i.e., $C_0 = (c_1, \dots, c_{I_0}) \sim \text{Ellip}(\mu_0, \Sigma_0, \phi)$ and $C_k = (c_1, \dots, c_{I_k}) \sim \text{Ellip}(\mu_k, \Sigma_k, \phi)$. Where μ_k is a mean vector and $\Sigma_k = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1I_k} \\ \vdots & \vdots & \dots & \vdots \\ \sigma_{I_k 1} & \sigma_{I_k 2} & \dots & \sigma_{I_k I_k} \end{bmatrix}$ is a positive definite matrix, $k = 1, \dots, K$.

Assumption 2. In this research, we consider only the elliptical distributions present in Table 2.

Using the theory of copula and Sklar's properties, a deterministic equivalent of (4) is given by [41]

$$\begin{aligned}
&\min_{t\in\mathbb{R}^M_{++}} \sum_{i=1}^{I_0}\mu_i\prod_{j=1}^{M}t_j^{a_{ij}},\\
&\text{s.t} \quad \sum_{i=1}^{I_k}\mu_i\prod_{j=1}^{M}t_j^{a_{ij}}+\phi^{-1}((\alpha^{y_k})^{\frac{1}{\theta}})\sqrt{\sum_{i=1}^{I_k}\sum_{l=1}^{I_k}\sigma_{il}\prod_{j=1}^{M}t_j^{a_{ij}+a_{lj}}}\le 1,\ k=1,\dots,K,\\
&\quad \prod_{k=1}^{K}y_k\ge\alpha,\ 0<y_k\le 1\ ,\ k=1,\dots,K,
\end{aligned} \tag{5}$$

where ϕ^{-1} is the quantile function of a standard elliptical distribution.

Problem (5) is not convex, we apply then a logarithmic transformation by setting $t_j = e^{r_j}$. Problem (5) becomes then,

$$\begin{aligned}
&\min_{r\in\mathbb{R}^M} \sum_{i=1}^{I_0}\mu_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\},\\
&\text{s.t} \quad \sum_{i=1}^{I_k}\mu_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\}+\phi^{-1}((\alpha^{y_k})^{\frac{1}{\theta}})\sqrt{\sum_{i\in I_k}\sum_{l\in I_k}\sigma_{il}\exp\left\{\sum_{j=1}^{M}(a_{ij}+a_{lj})r_j\right\}}\le 1,\ k=1,\dots,K,\\
&\quad \sum_{k=1}^{K}-\ln(y_k)\le-\ln(\alpha),\\
&\quad -y_k<0\ ,\ k=1,\dots,K,\\
&\quad y_k\le 1\ ,\ k=1,\dots,K.
\end{aligned} \tag{6}$$

Theorem 2. *Problem (6) is biconvex.*

Proof. The convexity on r is straightforward. We have for Gumbel–Hougaard copula that $\theta\ge 1$, it follows that $0<\frac{1}{\theta}\le 1$. Since $0<y_k\le 1$, we obtain $0<y_k^{\frac{1}{\theta}}\le 1$. We have then, $(\alpha^{y_k})^{\frac{1}{\theta}}\ge\alpha\ge 0.5$. From Table 2, $y_k\mapsto\phi^{-1}((\alpha^{y_k})^{\frac{1}{\theta}})$ is convex. The conclusion follows. □

We present in the rest of the section the optimality conditions of (6). First of all, we give the feasible set of (6) by

$$\begin{aligned}
\mathrm{U}=\Bigg\{ (r,y)\in\mathbb{R}^M\times\mathbb{R}^K \quad | \quad &\sum_{i\in I_k}\mu_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\}+\phi^{-1}((\alpha^{y_k})^{\frac{1}{\theta}})\sqrt{\sum_{i\in I_k}\sum_{l\in I_k}\sigma_{il}\exp\left\{\sum_{j=1}^{M}(a_{ij}+a_{lj})r_j\right\}}\le 1,\\
&0\le y_k\le 1\ ,\ k=1,\dots,K \text{ and } \sum_{k=1}^{K}-\ln(y_k)\le-\ln(\alpha)\Bigg\}.
\end{aligned}$$

We define additionally

$$\begin{aligned}
\mathrm{U(r)}=\Bigg\{ y\in\mathbb{R}^K \quad | \quad &\sum_{i\in I_k}\mu_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\}+\phi^{-1}((\alpha^{y_k})^{\frac{1}{\theta}})\sqrt{\sum_{i\in I_k}\sum_{l\in I_k}\sigma_{il}\exp\left\{\sum_{j=1}^{M}(a_{ij}+a_{lj})r_j\right\}}\le 1,\\
&0\le y_k\le 1\ ,\ k=1,\dots,K \text{ and } \sum_{k=1}^{K}-\ln(y_k)\le-\ln(\alpha)\Bigg\}.
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{U(y)}=\Bigg\{ r\in\mathbb{R}^M \quad | \quad &\sum_{i\in I_k}\mu_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\}+\phi^{-1}((\alpha^{y_k})^{\frac{1}{\theta}})\sqrt{\sum_{i\in I_k}\sum_{l\in I_k}\sigma_{il}\exp\left\{\sum_{j=1}^{M}(a_{ij}+a_{lj})r_j\right\}}\le 1,\\
&k=1,\dots,K\Bigg\}.
\end{aligned}$$

Definition 6. $(r^*,y^*)\in\mathbb{R}^M\times\mathbb{R}^K$ is a *partial optimum* of (6) if $\sum_{i\in I_0}\mu_i\exp\left\{\sum_{j=1}^{M}a_{ij}{r_j}^*\right\}\le\sum_{i\in I_0}\mu_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\}$, $\forall r\in\mathrm{U}(y^*)$ and $y^*\in\mathrm{U}(r^*)$

Definition 7. 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

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

$$\sum_{i\in I_k}\gamma_{y_k}\nabla_{y_k}g_k(r,y) + \beta_{+k} - \beta_{-k} - \frac{\lambda}{y_k} = 0, k = 1,\dots,K \tag{7}$$

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

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

then (r^*, y^*) is called a *partial KKT point* of (6).

Theorem 3. *Let $(r^*, y^*) \in \mathbb{R}^M \times \mathbb{R}^K$. If (6) is satisfied with partial Slater constraint qualification at (r^*, y^*), then (r^*, y^*) is a partial optimum of (6) if and only if (r^*, y^*) is a partial KKT point of (6). Furthermore if $\gamma_r = \gamma_y$, then (r^*, y^*) is a KKT point of (6).*

4. A neurodynamic approach

In this section, we present a dynamical neural network that converges to a solution of (6). We consider then two time-dependent variables $r(.)$ and $y(.)$ and the following dynamical equation

$$\begin{aligned}
\frac{dr_j}{dt} &= -\left(\sum_{i\in I_0}a_{ij}\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\} + \sum_{k=1}^{K}\left(\gamma_k + \left\{\sum_{i\in I_k}\mu_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\} + g_k(r,y)\right\}\right)_+\right. \\
&\quad \left.\times\left(\sum_{i\in I_k}a_{ij}\mu_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\} + \nabla_{r_j}g_k(r,y_k)\right)\right), j = 1,\dots,M, \\
\frac{dy_k}{dt} &= -\left(\sum_{i\in I_k}\nabla_{y_k}g_k(r,y)\times\left(\gamma_k + \left\{\sum_{i\in I_k}\mu_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\} + g_k(r,y)\right\}\right)_+\right. \\
&\quad \left. + \beta_{+k}(\beta_{+k} + y_k - 1)_+ + \beta_{-k}(\beta_{-k} - y_k)_+ - \frac{\lambda}{y_k}\left(\lambda + \ln(\alpha) - \sum_{k=1}^{K}\ln(y_k)\right)_+\right), k = 1,\dots,K, \\
\frac{d\gamma_k}{dt} &= \left(\gamma_k + \left\{\sum_{i\in I_k}\mu_i\exp\left\{\sum_{j=1}^{M}a_{ij}r_j\right\} + g_k(r,y)\right\}\right)_+ - \gamma_k, k = 1,\dots,K, \qquad \text{(DNN)} \\
\frac{d\beta_{+k}}{dt} &= \left(\beta_{+k} + y_k - 1\right)_+ - \beta_{+k}, k = 1,\dots,K, \\
\frac{d\beta_{-k}}{dt} &= \left(\beta_{-k} - y_k\right)_+ - \beta_{-k}, k = 1,\dots,K, \\
\frac{d\lambda}{dt} &= \left(\lambda + \ln(\alpha) - \sum_{k=1}^{K}\ln(y_k)\right)_+ - \lambda.
\end{aligned}$$

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

$$\frac{dr}{dt} = A(r,y,\gamma), \tag{8}$$
$$\frac{dy}{dt} = B(r,y,\gamma,\beta_+,\beta_-,\lambda), \tag{9}$$
$$\frac{d\gamma}{dt} = C(r,y,\gamma), \tag{10}$$
$$\frac{d\beta_+}{dt} = D(y,\beta_+), \tag{11}$$
$$\frac{d\beta_-}{dt} = E(y,\beta_-), \tag{12}$$
$$\frac{d\lambda}{dt} = F(y,\lambda). \tag{13}$$

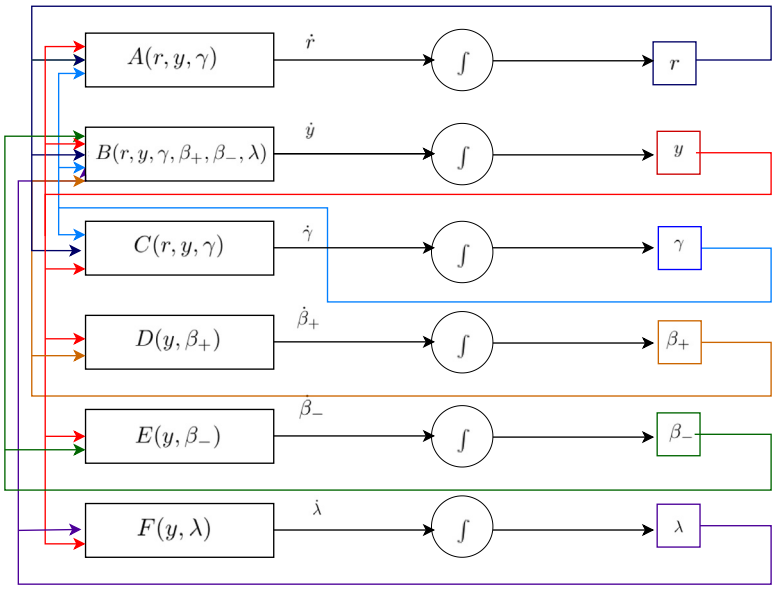

Fig. 1. A simplified implementation of the neural network (8)–(13).

For ease of understanding, we provide a pseudocode in Algorithm 1 and a simplified circuit implementation of Eqs. (8)–(13) in Fig. 1. Where, the symbol $\int$ represents the integration operator for reference.

Algorithm 1 A neurodynamic algorithm

Step 1. Select an arbitrary starting point, denoted as $z_0 = (r_0, y_0, \gamma_0, \beta_{+0}, \beta_{-0}, \lambda_0)$.
Step 2. Initialize the iteration counter $i = 0$.
Step 3. Update $z_{i+1} = z^*$, where z^* is an equilibrium point of the dynamical system (8)-(13) with the starting point z_i.
if A stopping criterion is satisfied, i.e, $||z_{i+1} - z_i|| \leq \epsilon$ **then**
STOP.
else
$i = i + 1$.
end if

Theorem 4. *Suppose that (r^*, y^*) is a partial optimum of (6) and $\gamma^*, \beta_+^*, \beta_-^*$ and λ^* the corresponding Lagrange multipliers, then $(r^*, y^*, \gamma^*, \beta_+^*, \beta_-^*, \lambda^*)$ is an equilibrium point of (DNN). Additionally, every equilibrium point of (DNN) is a KKT point of (6).*

Proof. Let $(r^*, y^*, \gamma^*, \beta_+^*, \beta_-^*, \lambda^*)$ an equilibrium point of (DNN), then all the left side derivatives in (DNN) are equal to zero. We use the fact that $\big((a+b)_+ = a\big) \Leftrightarrow \big(a \geq 0, b \leq 0 \text{ and } a^T b = 0\big)$ and we obtain the partial KKT system (7) with $\gamma_r = \gamma_y = \gamma$. Now if (r^*, y^*) is a partial optimum of (6) and $\gamma^*, \beta_+^*, \beta_-^*$ and λ^* the corresponding Lagrange multipliers, then it is straightforward that $(r^*, y^*, \gamma^*, \beta_+^*, \beta_-^*, \lambda^*)$ is an equilibrium point of (DNN). □

To establish the convergence and the stability for the neural network proposed to solve (4), we first provide the following definition and lemma.

Definition 8. A function $F : \mathbb{R}^m \to \Omega \subset \mathbb{R}^m$ is said to be monotone on Ω, if for each $x, y \in \Omega$

$$(F(x) - F(y))^T (x - y) \geq 0.$$

Lemma 1. *If the Jacobian matrix of a function F is positive semidefinite, then F is monotone. [42]*

Theorem 5. *The neurodynamic model (DNN) is stable and globally convergent to a KKT point $(r^*, y^*, \gamma^*, \beta_+^*, \beta_-^*, \lambda^*)$ of (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 Lemma 3 in [34], 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

$$\frac{dV(\mu)}{dt} = \underbrace{\Phi^T(\nabla\Phi(\mu)^T + \nabla\Phi(\mu))\Phi}_{\leq 0 \text{ since } \nabla\Phi \text{ is negative semidefinite}} + \underbrace{(\mu-\mu^*)^T(\Phi(\mu)-\Phi(\mu^*))}_{\leq 0 \text{ by Lemma 1}}.$$

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 problem (6). □

5. Numerical experiments

In this section, we consider three geometric problems to evaluate the performance of our neurodynamic approach. All the computations were performed using Python on an Intel Core i7-10610U CPU. The random data was generated using numpy.random and the solution of ODE systems was obtained using the function solve_ivp of scipy.integrate. The gradients and partial derivatives were computed using autograd.grad and autograd.jacobian. We only account for the quality of the solution and do not record the CPU time as current ODE solvers are time consuming.

5.1. Example 1 : A posynomial model of inductors

When dealing with the optimization of power electronic converters for multiple objectives, such as power density and efficiency, optimizing the magnetic components can be particularly challenging and time-consuming. To expedite and streamline the optimization process, it would be beneficial to formulate converter optimization as a geometric program. Stupar et al. [43] formulate the optimization of inductors for multiple design objectives as a geometric program. The resulting program is given by.

$$\begin{aligned} \min\ & \gamma\frac{\mathrm{P_L}}{\mathrm{P_{L,\ max}}} + (1-\gamma)\frac{\mathrm{Vol_L}}{\mathrm{Vol_{L,max}}}, \\ \text{s.t}\ & \mathrm{T_{core}} \leq \mathrm{T_{max}}, \\ & \mathrm{T_{winding}} \leq \mathrm{T_{max}}, \\ & \mathrm{Sat} \leq 1. \end{aligned} \tag{14}$$

where,

$$\begin{aligned} \mathrm{P_L} &= \alpha_1 \mathrm{L}^{1.1813}\mathrm{N}^{0.0718}(1000 l_g)^{-1.0063}\Delta\mathrm{I_L}^{2.3202}\mathrm{I_L}^{0.2450} f^{1.0821}\mathrm{F_F}^{-0.0343} \\ &\quad + \alpha_2 \mathrm{L}^{-0.5766}\mathrm{N}^{1.1049}(1000 l_g)^{0.2825}\Delta\mathrm{I_L}^{0.4946}\mathrm{I_L}^{1.2980} f^{-0.6408}\mathrm{F_F}^{-0.4949}, \\ \mathrm{Vol_L} &= \alpha_3 \mathrm{L}^{-0.7187}\mathrm{N}^{0.5152}(1000 l_g)^{-0.5337}\Delta\mathrm{I_L}^{-0.1637}\mathrm{I_L}^{-0.0107} f^{-0.5213}\mathrm{F_F}^{1.2685} \\ &\quad + \alpha_4 \mathrm{L}^{1.1614}\mathrm{N}^{-1.9220}(1000 l_g)^{0.7552}\Delta\mathrm{I_L}^{0.2451}\mathrm{I_L}^{0.1116} f^{0.2422}\mathrm{F_F}^{0.0446}, \\ \mathrm{T_{core}} &= \alpha_5 \mathrm{L}^{-5.1598}\mathrm{N}^{1.3502}(1000 l_g)^{-0.5282}\Delta\mathrm{I_L}^{-4.8747}\mathrm{I_L}^{-1.4170} f^{-4.5997}\mathrm{F_F}^{0.0413} \\ &\quad + \alpha_6 \mathrm{L}^{0.6982}\mathrm{N}^{-0.2495}(1000 l_g)^{1.2176}\Delta\mathrm{I_L}^{0.0256}\mathrm{I_L}^{1.4824} f^{0.6125}\mathrm{F_F}^{-0.1169}, \\ \mathrm{T_{winding}} &= \alpha_7 \mathrm{L}^{0.4714}\mathrm{N}^{-0.0047}(1000 l_g)^{-0.0695}\Delta\mathrm{I_L}^{0.6719}\mathrm{I_L}^{0.6318} f^{0.4644}\mathrm{F_F}^{0.0202} \\ &\quad + \alpha_8 \mathrm{L}^{-2.1699}\mathrm{N}^{2.0525}(1000 l_g)^{0.3287}\Delta\mathrm{I_L}^{-1.1485}\mathrm{I_L}^{0.3329} f^{-1.8299}\mathrm{F_F}^{-0.4741}, \\ \mathrm{Sat} &= \alpha_9 \mathrm{L}^{0.3952}\mathrm{N}^{-1.3325}(1000 l_g)^{-0.1931}\Delta\mathrm{I_L}^{-0.3167}\mathrm{I_L}^{-0.4861} f^{-1.4066}\mathrm{F_F}^{-0.3525} + 1, \end{aligned}$$

where $\mathrm{P_L}$ is the total losses, $\mathrm{Vol_L}$ is the boxed volume, $\mathrm{T_{max}}$ is the maximum allowable temperature of the inductor, $\mathrm{T_{core}}$ is the core temperature, $\mathrm{T_{winding}}$ is the winding temperature and Sat is the saturation condition. The design variables are the inductance L, the number of turns N, the air gap l_g, the current ripple $\Delta\mathrm{I_L}$, the average DC current $\mathrm{I_L}$, the frequency f and the fill factor $\mathrm{F_F}$. The normalization factor $\mathrm{P_{L,\ max}}$, which represents the maximum possible losses within the given design space, is obtained by solving (14) while considering only the volume (i.e., setting $\gamma = 0$). Similarly, the normalization factor $\mathrm{Vol_{L,max}}$, which represents the maximum possible volume within the given design space, is derived by solving (14) while considering only the losses (i.e., setting $\gamma = 1$).

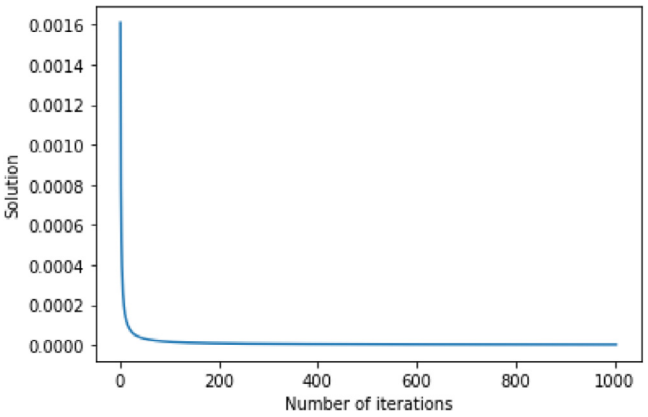

Fig. 2. Convergence of the optimal solution.

Table 3
Results for different values of γ and θ.

γ	$\theta = 1$ (independent constraints)		$\theta = 2$		$\theta = 10$	
	Obj value	VS	Obj value	VS	Obj value	VS
0.0	$2,0.10^{-6}$	5	$1,5.10^{-6}$	8	$1,4.10^{-6}$	10
0.5	$7,1.10^{-5}$	3	$6.8.10^{-5}$	12	$6,9.10^{-5}$	15
1.0	$8,7.10^{-5}$	4	$7,2.10^{-5}$	9	$5,9.10^{-6}$	16

We assume in our case that $\alpha_1, \alpha_2, \dots, \alpha_7$ are dependent random variables. We rewrite then (14) as follows

$$\begin{aligned} &\min\ \gamma \frac{\bar{\mathrm{P}}_{\mathrm{L}}}{\mathrm{P}_{\mathrm{L,\ max}}} + (1-\gamma)\frac{\bar{\mathrm{Vol}}_{\mathrm{L}}}{\mathrm{Vol}_{\mathrm{L,max}}}, \\ &\text{s.t}\quad \mathbb{P}\left(\mathrm{T}_{\mathrm{core}} \le \mathrm{T}_{\mathrm{max}}, \mathrm{T}_{\mathrm{winding}} \le \mathrm{T}_{\mathrm{max}}, \mathrm{Sat} \le 1\right) \ge 1-\epsilon. \end{aligned} \tag{15}$$

For the numerical experiments, we set $\mathrm{P}_{\mathrm{L,\ max}} = 10$ W, $\mathrm{Vol}_{\mathrm{L,max}} = 10\ \mathrm{m}^3$, $\mathrm{T}_{\mathrm{max}} = 50\ ^\circ$C, the mean values of α_i, $i = 1, \dots, 7$ are equal to the deterministic values given in [43]. For the sake of simplicity, we set the values of the diagonal coefficients of the covariance matrix to 0.1 and the remaining coefficients to 0.05. We assume that the variables α_i, $i = 1, \dots, 7$ are normally distributed and that their dependence is driven by Gumbel–Hougaard copula. We first solve (15) for $\gamma = 0$ $\epsilon = 5\%$ and $\theta = 5$, we follow the convergence of the optimal solution in Fig. 2. We observe that the final loss converges to 0. We fix now $\epsilon = 10\%$ and we solve (15) for different values of γ and θ. We test the robustness of the different solutions by generating 100 random samples of the variables α_i, $i = 1, \dots, 7$ using the function numpy.random.multivariate_normal. Then we count the number of times over 100 when one of the constraints of (15) was not satisfied. We call every counted time a violated scenario (VS). The obtained results are shown in Table 3. First column gives the value of γ. Second and third columns give the obtained objective value and the number of VS when $\theta = 1.0$, respectively. Columns four and five show the obtained objective value and the number of VS when $\theta = 2.0$, respectively. Finally, columns six and seven present the final value and the number of VS when $\theta = 1.0$, respectively. We observe that as θ increases, the number of violated scenarios increases.

5.2. Example 2 : A shape optimization problem

To assess the performance of our dynamical neural network, we employed the multidimensional shape optimization problem with joint chance constraints from [44].

$$\begin{aligned} &\min_{x \in \mathbb{R}^M_{++}} \prod_{i=1}^{m} x_i^{-1}, \\ &\text{s.t}\quad \mathbb{P}\left(\sum_{j=1}^{m-1}\left(\frac{m-1}{A_{wall\,j}} x_1 \prod_{i=1, i\neq j}^{m} x_i\right), \frac{1}{A_{floor}} \prod_{j=2}^{m} x_j \le 1\right) \ge 1-\epsilon, \\ &\quad \frac{1}{\gamma_{i,j}} x_i x_j^{-1} \le 1, 1 \le i \ne j \le m. \end{aligned} \tag{16}$$

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 $j = 1, \dots, m$ were considered as dependent

Table 4
Neural network vs. the sequential algorithm for normal distribution.

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
Results for different values of m 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 6
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	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 7
Results for different values of m for Logistic 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

variables. The mean value of $\frac{1}{A_{floor}}$ was set to 1.0/20.0 and the mean values of $\frac{1}{A_{wall_j}}$, $j = 1, \dots, m$ were randomly selected from the range $[1.0/60.0, 1.0/40.0]$. For the sake of simplicity, the values 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 [35] 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 (16) for all 100 cases. If the solutions are not feasible for a particular case, it is referred to as a violated scenario (VS). The numerical results are displayed in Table 4. 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 $= \frac{(\text{Obj value}_{\text{SA}} - \text{Obj value}_{\text{NN}})}{\text{Obj value}_{\text{SA}}} \times 100$, where Obj value_{NN} and Obj value_{SA} represent the objective values obtained by the neural network and sequential algorithm, respectively. Table 4 indicates that our neural network outperforms the sequential algorithm, as the upper bounds obtained are better and fewer scenarios are violated.

We solve (16) for different values of m and different values of θ; the dependence parameter of Gumbel–Hougaard Copula. Tables 5, 6, 7 show the obtained results 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.

Table 8
Neural network vs. the sequential algorithm for normal distribution.

K	θ	Neural network	VS	Sequential algorithm	VS	GAP
5	1	24.91	5	25.14	11	0.9
5	2	25.43	4	25.63	9	0.78
5	10	26.43	2	26.62	8	0.71
10	10	61.58	11	61.95	16	0.59
10	2	62.23	5	62.53	14	0.47
10	10	63.55	1	63.73	11	0.28
15	2	76.59	6	76.63	11	0.05
15	10	83.01	1	83.08	8	0.08
20	10	106.98	13	107.11	15	0.12
20	2	122.67	3	122.76	8	0.07
20	10	136.58	0	136.65	7	0.05

5.3. Example 3 : maximizing the worst user signal to interference noise ratio

In this subsection, our focus is on optimizing the maximum Signal to Interference Noise Ratio (SINR) for users in Massive Multiple Input Multiple Output (MaMIMO) systems from [45]. The optimization problem involves selecting a subset of antennas from a larger set, while ensuring that the interference in the system remains within a maximum limit. We consider a single cell area composed of a set of users $\mathcal{U} = 1, \dots, K$, with each user using only one antenna to receive data from the base station. The base station has T antennas and the aim is to maximize the worst user signal-to-interference noise ratio (SINR) while limiting the power assigned to each user. The optimization problem is formulated to maximize the minimum SINR of all users subject to constraints on the power assigned to each user and given by

$$\max_{p\in\mathbb{R}^K_{++}} \min_{i\in\mathcal{U}} \frac{p_i|g_i^H g_i|^2}{\sum_{j\in\mathcal{U},j\neq i} p_j|g_i^H g_j|^2 + |\sigma_i|^2}, \tag{17}$$

$$\text{s.t} \quad P_{min} \leq p_i \leq P_{max}, \forall i \in \mathcal{U}, \tag{18}$$

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.

Taking $a_{ij} = |g_i^H g_j|^2 |g_i^H g_i|^{-2}$ and $b_i = |\sigma_i|^2 |g_i^H g_i|^{-2}$, a geometric reformulation of (17)–(18) is given by

$$\min_{p\in\mathbb{R}^K_{++}, w\in\mathbb{R}_{++}} \quad w^{-1}, \tag{19}$$

$$\text{s.t} \quad \sum_{j\in\mathcal{U},j\neq i} a_{ij} p_j p_i^{-1} w + b_i p_i^{-1} w \leq 1, \forall i \in \mathcal{U}, \tag{20}$$

$$P_{min} \leq p_i \leq P_{max}, \forall i \in \mathcal{U}. \tag{21}$$

We assume that the coefficients a_{ij} and b_i are dependent and normally distributed and that the dependence is driven by Gumbel–Hougaard copula. As a result, we replace (19)–(21) by the following joint optimization problem

$$\min_{p\in\mathbb{R}^K_{++}, w\in\mathbb{R}_{++}} \quad w^{-1}, \tag{22}$$

$$\text{s.t} \quad \mathbb{P}\left\{ \sum_{j\in\mathcal{U},j\neq i} a_{ij} p_j p_i^{-1} w + b_i p_i^{-1} w \leq 1, \forall i \in \mathcal{U} \right\} \geq 1 - \epsilon, \tag{23}$$

$$P_{min} \leq p_i \leq P_{max}, \forall i \in \mathcal{U}. \tag{24}$$

For the numerical experiments, we set $P_{min} = 0.1$, $P_{max} = 0.5$ and $\epsilon = 0.15$. The mean vectors are uniformly generated in $[3.0, 7.0]$, the diagonal coefficients of the covariance matrices are uniformly taken in $[0.1, 0.3]$ and the remaining coefficients are set to 0.1. The results are presented in Table 8. Column one gives the value of K. Column two shows the value of the dependency parameter θ. Columns three and four give the optimal solution value and the number of VS for the dynamical neural network, respectively. Columns five and six present the optimal solution value and the number of VS for the sequential algorithm, respectively. Finally, column seven gives the value of the gap between the two obtained solutions. We observe that the dynamical neural network gives better solutions compared to the sequential algorithm and ensures better robustness. Fig. 3 shows that the outperforming of the dynamical neural networks is ensured for different values of ϵ and θ.

6. Conclusion

This study introduces a new neurodynamic approach to solve joint chance-constrained geometric programming problems with dependent row vectors based on Copula theory. The joint chance constraint was first transformed into an equivalent deterministic constraint. Then a standard variable transformation was used to derive a biconvex reformulation of the deterministic optimization

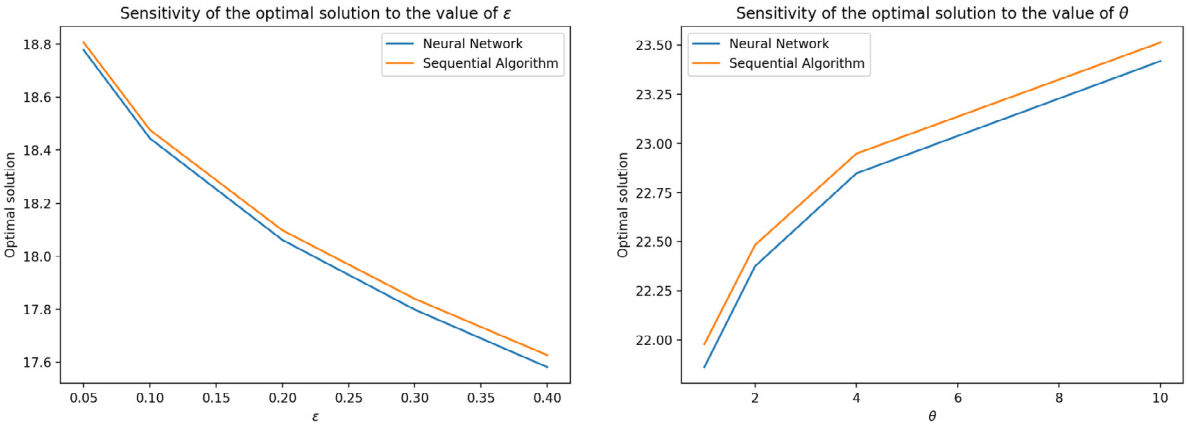

Fig. 3. Sensitivity of the optimal solution to the values of ϵ and θ.

model. Later, based on the optimality conditions of the obtained deterministic equivalent we construct a recurrent neural network to approximate the solution of the stochastic initial problem.

One of the major benefits of this research is the ability to solve dependent joint chance-constrained geometric programs without using any convex or linear approximation techniques. Three numerical examples are presented in the numerical Section to show the quality of the proposed method. The results indicate that our approach approximates well the optimal solution compared to the state-of-the-art existing methods and covers well the risk area by providing robust solutions.

Our innovative approach using dynamical neural networks can be applied to a wide range of optimization and game theory problems. Specifically, it can handle various types of constrained nonlinear optimization problems, including convex smooth and nonsmooth optimization, pseudo-convex nonsmooth optimization, variational inequalities, nonlinear complementarity problems, quadratic optimization, dynamic optimization, as well as stochastic and distributionally robust optimization problems. In the field of game theory, our approach enables the computation of Nash equilibrium for finite n-player chance constrained games and differential games. Additionally, we foresee the potential application of our dynamical neural network in chance constrained Markov Decision Processes.

However, it is important to acknowledge that the current version of the algorithm is time-consuming due to the iterative solutions of the dynamical differential system that describes the model. Nevertheless, there are opportunities to enhance the algorithm's efficiency and quality through further research and development. One approach can be improved by implementing ODE (Ordinary Differential Equation) solvers based on artificial intelligence techniques. AI techniques such as neural networks or reinforcement learning can potentially improve the speed and accuracy of solving the dynamical differential system. Additionally, other computing techniques, e.g., parallel computing, GPU acceleration, or distributed computing can be employed to speed up the algorithm's execution time further. These approaches can take advantage of hardware advancements to process computations in parallel, reducing the overall time needed for solving the system.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Charnes A, Cooper WW. Chance-constrained programming. Manage Sci 1959;6(1):73–9, URL https://EconPapers.repec.org/RePEc:inm:ormnsc:v:6:y:1959:i:1:p:73-79.

[2] Andrieu L, Henrion R, Römisch W. A model for dynamic chance constraints in hydro power reservoir management. European J Oper Res 2010;207(2):579–89. http://dx.doi.org/10.1016/j.ejor.2010.05.013, URL https://www.sciencedirect.com/science/article/pii/S0377221710003693.

[3] Cetinkaya E, Thiele A. Data-driven portfolio management with quantile constraints. 2014, http://dx.doi.org/10.1007/s00291-015-0396-9.

[4] Choi B-G, Rujeerapaiboon N, Jiang R. Multi-period portfolio optimization: Translation of autocorrelation risk to excess variance. Oper Res Lett 2016;44. http://dx.doi.org/10.1016/j.orl.2016.10.006.

[5] Deng Y, Shen S. Decomposition algorithms for optimizing multi-server appointment scheduling with chance constraints. Math Program 2016;157. http://dx.doi.org/10.1007/s10107-016-0990-x.

[6] Wang S, Li J, Mehrotra S. A solution approach to distributionally robust joint-chance-constrained assignment problems. INFORMS J Optim 2022;4(2):125–47. http://dx.doi.org/10.1287/ijoo.2021.0060, arXiv:https://doi.org/10.1287/ijoo.2021.0060.

[7] Ma S, Sun D. Chance constrained robust beamforming in cognitive radio networks. IEEE Commun Lett 2013;17(1):67–70. http://dx.doi.org/10.1109/LCOMM.2012.112812.121829.

[8] Lejeune MA, Shen S. Multi-objective probabilistically constrained programs with variable risk: Models for multi-portfolio financial optimization. European J Oper Res 2016;252(2):522–39. http://dx.doi.org/10.1016/j.ejor.2016.01.03, URL https://ideas.repec.org/a/eee/ejores/v252y2016i2p522-539.html.

[9] Nemirovski A, Shapiro A. Convex approximations of chance constrained programs. SIAM J Optim 2007;17(4):969–96. http://dx.doi.org/10.1137/050622328, arXiv:https://doi.org/10.1137/050622328.

[10] Chen W, Sim M, Sun J, Teo C-P. From CVaR to uncertainty set: Implications in joint chance-constrained optimization. Oper Res 2010;58(2):470–85. http://dx.doi.org/10.1287/opre.1090.0712, arXiv:https://doi.org/10.1287/opre.1090.0712.

[11] Cheng J, Lisser A. A second-order cone programming approach for linear programs with joint probabilistic constraints. Oper Res Lett 2012;40(5):325–8. http://dx.doi.org/10.1016/j.orl.2012.06.008.

[12] Tassouli S, Lisser A. Solving linear programs with joint probabilistic constraints with dependent rows using a dynamical neural network. Results Control Optim 2022;9:100178. http://dx.doi.org/10.1016/j.rico.2022.100178, URL https://www.sciencedirect.com/science/article/pii/S2666720722000509.

[13] Zadeh L. Fuzzy sets. Inf Control 1965;8(3):338–53. http://dx.doi.org/10.1016/S0019-9958(65)90241-X, URL https://www.sciencedirect.com/science/article/pii/S001999586590241X.

[14] Sharma K, Singh VP, Ebrahimnejad A, Chakraborty D. Solving a multi-objective chance constrained hierarchical optimization problem under intuitionistic fuzzy environment with its application. Expert Syst Appl 2023;217:119595. http://dx.doi.org/10.1016/j.eswa.2023.119595, URL https://www.sciencedirect.com/science/article/pii/S0957417423000969.

[15] Rani D, Ebrahimnejad A, Gupta G. Generalized techniques for solving intuitionistic fuzzy multi-objective non-linear optimization problems. Expert Syst Appl 2022;202:117264. http://dx.doi.org/10.1016/j.eswa.2022.117264, URL https://www.sciencedirect.com/science/article/pii/S0957417422006364.

[16] Peykani P, Lotfi F, Sadjadi S, Ebrahimnejad A, Mohammadi E. Fuzzy chance-constrained data envelopment analysis: A structured literature review, current trends, and future directions. Fuzzy Optim Decis Mak 2021. http://dx.doi.org/10.1007/s10700-021-09364-x.

[17] Kochenberger GA. Inventory models: Optimization by geometric programming. Decis Sci 1971;2(2):193–205. http://dx.doi.org/10.1111/j.1540-5915.1971.tb01454.x, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-5915.1971.tb01454.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-5915.1971.tb01454.x.

[18] Maranas CD, Floudas CA. Global optimization in generalized geometric programming. Comput Chem Eng 1997;21(4):351–69. http://dx.doi.org/10.1016/S0098-1354(96)00282-7, URL https://www.sciencedirect.com/science/article/pii/S0098135496002827.

[19] Rijckaert MJ, Martens XM. A condensation method for generalized geometric programming. Math Program 1976;11(1):89–93. http://dx.doi.org/10.1007/bf01580373.

[20] Bricker DL, Rajgopal J. Yet another geometric programming dual algorithm. Oper Res Lett 1983;2(4):177–80. http://dx.doi.org/10.1016/0167-6377(83)90051-2, URL https://www.sciencedirect.com/science/article/pii/0167637783900512.

[21] Kortanek K, Xu X, Ye Y. An infeasible interior-point algorithm for solving primal and dual geometric programs. Math Program 1996;76:155–81. http://dx.doi.org/10.1007/BF02614382.

[22] Mondal, Tapas, Ojha, Akshay Kumar, Pani, Sabyasachi. Solving geometric programming problems with triangular and trapezoidal uncertainty distributions. RAIRO-Oper Res 2022;56(4):2833–51. http://dx.doi.org/10.1051/ro/2022132.

[23] Liu S-T. Posynomial geometric programming with parametric uncertainty. European J Oper Res 2006;168(2):345–53. http://dx.doi.org/10.1016/j.ejor.2004.04.046, URL https://www.sciencedirect.com/science/article/pii/S037722170400390X. Feature Cluster on Mathematical Finance and Risk Management.

[24] Liu S-T. Geometric programming with fuzzy parameters in engineering optimization. Internat J Approx Reason 2007;46(3):484–98. http://dx.doi.org/10.1016/j.ijar.2007.01.004, URL https://www.sciencedirect.com/science/article/pii/S0888613X07000060. Special Section: Aggregation Operators.

[25] Shiraz RK, Fukuyama H. Integrating geometric programming with rough set theory. Oper Res 2018;18(1):1–32. http://dx.doi.org/10.1007/s12351-016-0250-0, URL https://ideas.repec.org/a/spr/operea/v18y2018i1d10.1007_s12351-016-0250-0.html.

[26] Shiraz RK, Tavana M, Fukuyama H, Caprio DD. Fuzzy chance-constrained geometric programming: the possibility, necessity and credibility approaches. Oper Res 2017;17(1):67–97. http://dx.doi.org/10.1007/s12351-015-0216-7, URL https://ideas.repec.org/a/spr/operea/v17y2017i1d10.1007_s12351-015-0216-7.html.

[27] Hosseini Nodeh Z, Azar A, Khanjani Shiraz R, Khodayifar S, Pardalos P. Joint chance constrained shortest path problem with copula theory. J Combin Optim 2020;40. http://dx.doi.org/10.1007/s10878-020-00562-8.

[28] Hopfield J, Tank D. Neural computation of decisions in optimization problems. Biol Cybern 1985;52:141–52. http://dx.doi.org/10.1007/BF00339943.

[29] Xia Y, Feng G. A new neural network for solving nonlinear projection equations. Neural Netw 2007;20(5):577–89. http://dx.doi.org/10.1016/j.neunet.2007.01.001, URL https://www.sciencedirect.com/science/article/pii/S0893608007000032.

[30] Liu N, Qin S. A neurodynamic approach to nonlinear optimization problems with affine equality and convex inequality constraints. Neural Netw 2019;109:147–58. http://dx.doi.org/10.1016/j.neunet.2018.10.010, URL https://www.sciencedirect.com/science/article/pii/S0893608018302958.

[31] Leung M-F, Wang J. A collaborative neurodynamic approach to multiobjective optimization. IEEE Trans Neural Netw Learn Syst 2018;29(11):5738–48. http://dx.doi.org/10.1109/tnnls.2018.2806481.

[32] Leung M-F, Wang J, Che H. Cardinality-constrained portfolio selection via two-timescale duplex neurodynamic optimization. Neural Netw 2022;153:399–410. http://dx.doi.org/10.1016/j.neunet.2022.06.023, URL https://www.sciencedirect.com/science/article/pii/S0893608022002386.

[33] Nazemi A, Tahmasbi N. A high performance neural network model for solving chance constrained optimization problems. Neurocomputing 2013;121:540–50. http://dx.doi.org/10.1016/j.neucom.2013.05.034, URL https://www.sciencedirect.com/science/article/pii/S0925231213006024. Advances in Artificial Neural Networks and Machine Learning.

[34] Tassouli S, Lisser A. A neural network approach to solve geometric programs with joint probabilistic constraints. Math Comput Simulation 2023;205:765–77. http://dx.doi.org/10.1016/j.matcom.2022.10.025, URL https://www.sciencedirect.com/science/article/pii/S0378475422004384.

[35] Khanjani Shiraz R, Khodayifar S, Pardalos P. Copula theory approach to stochastic geometric programming. J Global Optim 2021;81:1–34. http://dx.doi.org/10.1007/s10898-021-01062-7.

[36] Sklar A. Fonctions de répartition à n dimensions et leurs marges. Publ l'Inst Statist l'Univ Paris 1959;8:229–31.

[37] Jouini MN, Clemen RT. Copula models for aggregating expert opinions. Oper Res 1996;44(3):444–57, URL http://www.jstor.org/stable/171704.

[38] Patton A. Copula-based models for financial time series. OFRC working papers series, Oxford Financial Research Centre; 2008, http://dx.doi.org/10.1007/978-3-540-71297-8_34.

[39] Houda M, Lisser A. 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, Setubal, PRT: SCITEPRESS - Science and Technology Publications, Lda; 2014, p. 72–9. http://dx.doi.org/10.5220/0004831500720079.

[40] Liu Q, Yang S, Wang J. A collective neurodynamic approach to distributed constrained optimization. IEEE Trans Neural Netw Learn Syst 2017;28(8):1747–58. http://dx.doi.org/10.1109/TNNLS.2016.2549566.

[41] Cheng J, Houda M, Lisser A. Chance constrained 0–1 quadratic programs using copulas. Optim Lett 2015;9(7):1283–95. http://dx.doi.org/10.1007/s11590-015-0854-y.

[42] Ortega JM, Rheinboldt WC. Iterative solution of nonlinear equations in several variables. Society for Industrial and Applied Mathematics; 2000, http://dx.doi.org/10.1137/1.9780898719468, arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9780898719468. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898719468.

[43] Stupar A, Taylor JA, Prodic A. Posynomial models of inductors for optimization of power electronic systems by geometric programming. In: 2016 IEEE 17th workshop on control and modeling for power electronics. COMPEL, 2016, p. 1–8. http://dx.doi.org/10.1109/COMPEL.2016.7556660.

[44] Liu J, Lisser A, Chen Z. Distributionally robust chance constrained geometric optimization. Math Oper Res 2022;47(4):2950–88. http://dx.doi.org/10.1287/moor.2021.1233, arXiv:https://doi.org/10.1287/moor.2021.1233.

[45] Adasme P, Lisser A. A stochastic geometric programming approach for power allocation in wireless networks. Wirel Netw 2023. http://dx.doi.org/10.1007/s11276-023-03295-8.