Adding Copula Families to Copula-GP

Emmy Zhou

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2023

Abstract

Modelling the relationships between neuronal activity, sensory stimuli, and behavioural outputs is a key challenge in neuroscience. Within this field, the Copula-GP approach represents a significant advancement in the analysis of dynamic neural dependencies. However, its performance is limited when the true dependence structure deviates from that of the copula models employed in the framework. In this project, we address this limitation by incorporating three new parametric copula families (Ali-Mikhail-Haq, Joe, and Student's t copulas) to Copula-GP. We evaluate the performance of the extended Copula-GP framework on different datasets and assess their abilities in providing simple yet novel dependence structures. Our results demonstrate that incorporating the AMH and Student's t copulas significantly enhances the framework's expressive power, while the Joe copula's contribution is comparatively less significant. Furthermore, by extending the Copula-GP framework to accommodate multi-parameter copulas, our work makes a considerable advancement in enabling the inclusion of more complex dependence structures.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Emmy Zhou*)

Acknowledgements

I would like to extend my sincere gratitude to my supervisor, Dr. Arno Onken, for his unwavering support and kindness throughout this project. His guidance has been invaluable, and I am thankful for his willingness to address my many inquiries, including clarifying the correct plural form of copula.

I am also deeply grateful to Dr. Nina Kudryashova, who is the main creator behind Copula-GP. Her assistance in navigating me through the technical details of this project has been invaluable, and I am thankful for her eagerness in answering my many inquiries.

Lastly, I want to acknowledge my family and partner for their loving support during my time in Edinburgh. Their encouragement have been crucial to the successful completion of this project.

Table of Contents

Chapter 1

Introduction

1.0.1 Context and Motivation

Investigating neural dependence remains a challenging endeavour due to the highly dynamic and intricate nature of the brain. With billions of neurons, the brain features complex interactions and a wide range of response patterns that yield high-dimensional datasets, often with a limited number of trials [26]. State-of-the-art imaging and recording techniques have enabled researchers to simultaneously monitor large numbers of neurons as well as a variety of behavioural variables [22]. However, conventional measures of dependence are often inadequate to examine dependencies between such variables, as they can only capture linear or monotonic relationships. These methods also impose constraints on the marginal behaviour of the variables, even though neuronal spiking and behavioural variables often operate on substantially different timescales and exhibit unique marginal (i.e., single-variable) statistics. In particular, neuronal spiking takes place on a timescale of milliseconds, while behavioural variables range from seconds to hours or even days [22].

Copula models have emerged in recent years as a promising tool for modelling dependencies between variables with vastly different statistics [22]. By dissociating marginal statistics from the dependence structure, copula models enable a more comprehensive analysis of variable relationships. In addition, lower-dimensional copulas can be used to construct high-dimensional copulas through the so-called pair copula construction [2], effectively addressing the curse of dimensionality present in large datasets [26]. This ability to handle a variety of dependencies, including non-linear and high-dimensional ones, makes copula models particularly suitable for studying neuronal activity.

Within the field of computational neuroscience, copula models have been employed to study spiking activity between cortical neurons [17], spiking activity within the prefrontal cortex [30], and calcium transients in the visual cortex [26]. However, some of these models assume that the dependence between variables remains static, limiting their capacity to capture the dynamic nature of neural and behavioural interactions. To address this limitation, research efforts have focused on developing dynamic copula models that integrate continuous time- or context-dependent changes in the relationships between neuronal and behavioural variables. One such example is the Copula-GP framework

(Kudryashova et al.) which combines Gaussian processes and copula mixtures to dynamically model high-dimensional neuronal dependencies [22]. Specifically, the framework makes use of parametric conditional copulas that allow for modelling dependencies as they vary with respect to a task-related variable such as time, space, or orientation. This parametric approach was shown to significantly improve performance in characterising high-dimensional dependencies compared to other commonly-used techniques. Moreover, the Copula-GP model has proven useful in providing accurate mutual information estimates as well as predicting behaviorally relevant parameters of the task without the need for explicit cues.

The current iteration of the Copula-GP framework incorporates copula mixtures comprising four distinct parametric families: Gaussian, Clayton, Gumbel, and Frank. While the combinations of these families produce flexible copula mixtures with considerable expressive power, it has been observed that Copula-GP's performance deteriorates whenever the true dependence structure deviates from that of the parametric families [22]. Additionally, the current framework only considers single-parameter copulas, whereas other Gaussian process copula models have demonstrated promising results from accommodating copulas with multiple parameters [13]. As recent research has provided universal methods for constructing multi-parameter asymmetric copulas [24], the addition of multiple parameter copulas to Copula-GP represents a significant area of improvement.

1.0.2 Objectives

The main objective of this project is to improve the expressive power of the Copula-GP model [22] by adding more copula families. Specifically, we extend the framework to accommodate copulas with multiple parameters and consider the addition of three new parametric families: the Ali-Mikhail-Haq copula, the Joe copula, and the Student's t copula. The contribution of this project is twofold:

- We implement three new copula families within Copula-GP and evaluate their value to the framework.
- We propose a modified architecture for Copula-GP which allows for the inclusion of copulas with arbitrarily many parameters.

This project seeks to answer the following research questions:

RQ1: To what extent does the Ali-Mikhail-Haq copula contribute to the expressive power of the Copula-GP model?

RQ2: To what extent does the Joe copula contribute to the expressive power of the Copula-GP model?

RQ3: To what extent does the Student's t copula contribute to the expressive power of the Copula-GP model?

1.0.3 Thesis Structure

The remainder of this thesis is organised as follows:

- **Chapter 2** provides background on copula models, Gaussian processes, and the existing Copula-GP framework. It introduces the fundamental concepts and terminology necessary to understand the project's objectives and the proposed modifications to the Copula-GP framework.
- **Chapter 3** presents the methodology for implementing the Ali-Mikhail-Haq, Joe, and Student's *t* copulas into the Copula-GP framework. It also outlines the proposed modifications to the architecture, which enable the accommodation of copulas with multiple parameters.
- **Chapter 4** describes the experimental setup used to evaluate the performance of the extended Copula-GP framework, including how the synthetic datasets were generated. This chapter also presents the results of these experiments.
- **Chapter 5** discusses the results of the experiments from Chapter 4 in the context of the project's objectives and research questions. This chapter also explores alternative methods and suggests potential avenues for future work.
- **Chapter 6** concludes the thesis by summarising the main findings.

Chapter 2

Background and Literature Review

This section introduces the relevant background knowledge used in this project. In Section 2.1 and Section 2.2, we describe copula theory and Gaussian processes, which are the two main components behind Copula-GP. Next, in Section 2.3, we outline how these are combined in the Copula-GP framework. Finally, in Section 2.4, we review prior research related to neural copula models and Gaussian process vine copulas.

2.1 Copulas

2.1.1 Bivariate Copulas

In probability theory, copulas are functions that describe the dependence structures between random variables in multivariate distributions. Named after the Latin word for "link" and "bond", copulas provide a way in which marginal (single variable) probability distributions can be "coupled" to form a joint distribution. Conversely, copulas also allow the decomposition of a multivariate distribution into its marginals and dependence structure. Unlike traditional measures of dependence, such functions can capture complex, non-linear, and non-monotonic relationships, making them a powerful tool for analysing dependencies in neuronal data with inherent variability [6].

More formally, a bivariate copula $C : [0,1]^2 \rightarrow [0,1]$ is a cumulative distribution function (CDF) defined on the unit square with uniform marginals [12]. In the 2-dimensional case, we consider a random vector (X,Y) and the marginal CDFs of each component $F_X = P(X \leq x)$ and $F_Y = P(Y \leq y)$. Applying the probability integral transform yields a random vector $(U,V) = (F_X(X), F_Y(Y))$ that has uniformly distributed marginals on the interval $[0,1]$. The copula of (X,Y) is defined as the joint CDF of (U,V): $C(u,v) = P(U \leq u, V \leq v)$.

Using copulas, we can separate a joint distribution into its marginals and dependence structure. This is formalised in **Sklar's Theorem**, which states that any multivariate joint distribution can be expressed in terms of its univariate marginal distributions and a copula [6]. In a bivariate setting, this is equivalent to:

$$F(x,y) = C(F_x(x), F_y(y)) \tag{2.1}$$

Sklar's theorem can also be written in terms of probability densities:

$$f(x,y) = c(F_x(x), F_y(y)) \times f_x(x) \times f_y(y) \tag{2.2}$$

where $f_i(x)$ is the probability density function of the i-th component and c is the copula density defined as $c(u,v) = \frac{\partial C(u,v)}{\partial u \partial v}$.

In practice, a copula is learnt from data using a two-stage estimation method [13]. The first step consists of learning the marginal distributions of each component by fitting univariate models. The second step is mapping the data to the unit square via the probability integral transform and then fitting the empirical copula.

2.1.1.1 Parametric Copula Families

Parametric copulas represent a subcategory of copulas that have a fixed functional form defined by a finite set of parameters. These parameters dictate the shape and strength of the dependence structure within the copula, as well as its behaviour around extreme values, also known as tail dependence [19]. Numerous bivariate copula families stem from a variety of parametric bivariate distributions. The Gaussian copula, for instance, is an elliptical copula derived from the elliptical Gaussian distribution. It has one parameter ρ that controls the strength of dependence and can be implicitly written as [28]:

$$C(u,v) = P(X \leq \Phi^{-1}(u), Y \leq \Phi^{-1}(v)) = \Phi_{X,Y}(\Phi^{-1}(u), \Phi^{-1}(v); \rho) \tag{2.3}$$

where Φ is the CDF of the standard normal, $\Phi_{X,Y}$ is the joint CDF of (X,Y) and ρ their covariance. Other examples of elliptical copulas include the Student-t copula, which is constructed using the multivariate Student-t distribution. Like all elliptical distributions, elliptical copulas exhibit full rotational symmetry [28].

Archimedean copulas are another special class of parametric copulas characterised by a unifying structure based on a generator function [19]. The generator function φ, also called the Archimedean generator, is a continuous, strictly decreasing, and convex function that maps the unit interval [0,1] to the non-negative real line [0,∞). Using the generator, all bivariate Archimedean copulas can be expressed in the form $C(u,v) = \varphi(\varphi^{-1}(u) + \varphi^{-1}(v))$ and most Archimedean copulas admit an explicit formula [28]. Due to their ease of construction and their many desirable properties, Archimedean copulas are widely used in various applications, including in Copula-GP.

Figure 2.1 illustrates some of the most commonly-used elliptical and Archimedean copulas.

Copula Mixtures. A useful property of copulas is that a mixture of copulas is also a copula. This means that if we want to model correlations that cannot be described by a single copula alone, we can combine them to form a linear mixture model [22]:

$$C(u,v) = \sum_{j=1}^{N} \phi_j(x) C_j(u,v) \tag{2.4}$$

where $\phi_j(x)$ is the concentration and C_j the copula function of the j-th copula.

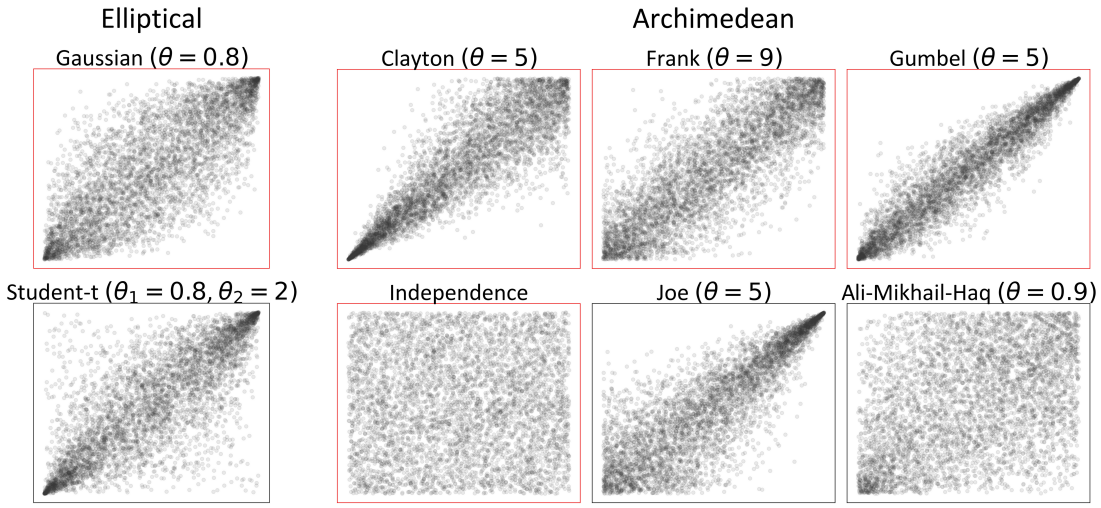

Figure 2.1: Samples drawn from commonly-used parametric copula families. The copula parameter θ effectively determines how the samples are distributed on the unit square. The copula families outlined in red are currently implemented in Copula-GP, and the copula families outlined in black will be implemented in this project.

2.1.1.2 Conditional Copulas

Conditional copulas are an extension of the standard copula framework that allows for the modelling of non-static dependence structures. In particular, conditional copulas capture the changing nature of dependence among random variables as a function of some continuous conditioning variable. This flexibility can be particularly valuable in situations where the dependence structure is not constant across different regions of the conditioning variable. One such case is in a neuronal experimental setting, where the dependence between neurons may vary as the stimulus is presented. In Copula-GP, conditional copulas play a vital role by dynamically modelling the dependence structure between neurons or behavioural variables, conditioned on some continuous task-related variable such as time, velocity, or orientation [22].

To derive the conditional copula, we first consider the standard bivariate copula $C(u,v)$, which is the joint cumulative distribution function (CDF) of U and V. Our goal is to condition on one of the variables, for instance, U, and obtain the conditional distribution function for V given $U = u$.

We denote the conditional copula as $C(v|u)$, which describes the conditional CDF of V when $U = u$. Using the definition of conditional probability, we can express the conditional copula as a partial derivative of $C(u,v)$ [28]:

$$C(v|u) = P(V \leq v|U = u) = \lim_{\Delta u \to 0} \frac{C(u+\Delta u, v) - C(u,v)}{\Delta u} = \frac{\partial C(u,v)}{\partial u} \tag{2.5}$$

The above method of differentiating with respect to the conditioning variable works well for copulas that admit an explicit formula. In cases where the copula function is implicitly defined, an alternative approach is to first obtain the conditional distribution

of the underlying variables, $F(y|x)$, and then use the probability integral transform to obtain the conditional copula.

2.1.1.3 Sampling from a Copula

Apart from modelling dynamic dependence structures, conditional copulas also offer a means to generate random variates from a copula. In the bivariate case, we can sample a pair (u,v) of observations of uniform $(0,1)$ random variables (U,V) whose joint distribution function is C using the conditional distribution method [28]:

1. Draw two samples u and t independently from the standard uniform distribution.
2. Let $v = C^{-1}(t|u)$, where $v = C^{-1}(t|u)$ is the quasi-inverse of $C(t|u)$.
3. The pair (u,v) follows the joint distribution function C.

2.1.1.4 Limitations of Bivariate Copulas

Although the aforementioned copulas are useful for modelling the dependence between two random variables, directly extending them to higher dimensions may not be suitable due to their limited flexibility in higher-order scenarios. Such extensions often fail to accurately model complex and varying dependence structures in higher dimensions, where capturing dependencies across all possible variable pairs becomes increasingly challenging and prone to oversimplification and misrepresentation. This is especially true in a neuronal setting, where dependencies between multiple neurons and behavioural variables are intricate and cannot be accurately modelled using a single copula family [26].

Furthermore, the curse of dimensionality can pose significant challenges for copula modelling in large neuronal datasets, as parameter estimation and model selection become more difficult with an increasing number of variables and a limited number of trials [26]. Consequently, directly extending bivariate copulas to the multivariate setting may result in suboptimal models that inadequately describe the relationships between variables. To address these limitations, more advanced and flexible methods, such as vine copulas, have been developed.

2.1.2 Vine Copulas

Vine copulas are a flexible class of multivariate copulas that offer a versatile way to model the dependence structure among more than two random variables. Building on the basic concepts of bivariate copulas, vine copulas extend the applicability of copulas to higher dimensions by constructing a hierarchical network of bivariate copulas, capturing complex interdependencies between multiple variables in a tractable manner [2].

2.1.2.1 Pair Copula Constructions

The foundation of vine copulas is the pair copula construction (PCC) approach, which decomposes the multivariate joint distribution into a product of pairwise copulas [4]. This construction is made possible by the use of conditional copulas, as discussed earlier,

to capture the dependencies between pairs of variables conditioned on other variables. In the PCC framework, a d-dimensional joint probability density $f(x_1, x_2, ..., x_d)$ can be represented as a product of $d(d-1)/2$ bivariate copula densities and d marginal densities [22].

To illustrate this decomposition process, let us consider the trivariate case (d=3) where the joint density $f(x_1, x_2, x_3)$ can be factored using the product rule of probability as $f(x_3) \cdot f(x_2|x_3) \cdot f(x_1|x_2, x_3)$. Rearranging Equation 2.2, we can express $f(x_2|x_3)$ as $c_{23}(F_2(x_2), F_3(x_3)) \cdot f_2(x_2)$, where $f_2(x_2)$ is the marginal density of x_2, and c_{23} is the copula of x_2 and x_3. Similarly, we can write $f(x_1|x_2, x_3)$ as $c_{12|3}(F_{1|3}(x_1|x_3), F_{2|3}(x_2|x_3)) \cdot f(x_1|x_3)$, where $f(x_1|x_3)$ can be further decomposed as before. This factorisation results in a cascade of conditional bivariate copulas, which is referred to as the pair copula construction.

From the example above, it is evident that the pair copula construction is inherently iterative and has many alternative re-parametrisations for any given factorisation. In fact, the number of potential constructions can be quite substantial, with 240 distinct constructions when d is only five [2]. To facilitate the organisation of these constructions, Bedford and Cooke [4] proposed a graphical model called the regular vine (R-vine). In this section, we will focus on one specific type of regular vines used in Copula-GP, namely the canonical vine or the C-vine.

2.1.2.2 Canonical Vine Copulas

C-vines, or canonical vines, are a specific class of vine copulas that possess a sequential tree structure. In a C-vine, the first tree links all variables to a single, fixed variable referred to as the root node. This root node does not change for all the trees in the C-vine. Each successive tree in the vine connects the remaining conditioning variables in a pairwise manner. The process continues until all the conditioning variables are exhausted. This tree structure leads to a natural and interpretable decomposition of the copula probability density function given by [22]:

$$c(\mathbf{u}) = \left[\prod_{i=2}^{N} c_{1i}(u_1, u_i)\right] \times \left[\prod_{i=2}^{N} \prod_{j=i+1}^{N} c_{ij|\{k\}_{k<i}}\Big(F(u_i|\{u_k\}_{k<i}), F(u_j|\{u_k\}_{k<i})\Big)\right] \quad (2.6)$$

2.2 Gaussian Processes

This section provides an overview of Gaussian Processes, which are used in Copula-GP to parameterise conditional copulas as they vary with respect to a task-related variable. Gaussian Processes are an ideal modeling choice for neuronal analysis, as they provide a Bayesian approach to learning. Given that neural spike trains frequently encompass a degree of inherent randomness, our interest often extend beyond that of fitting a single function to data. Instead, we want to consider a distribution of functions, which is precisely what Gaussian Processes can achieve.

A Gaussian Process (GP) is a high-dimensional multivariate Gaussian distribution over functions. Formally, it is defined as a collection of random variables, where any finite set of them have joint Gaussian distributions [34].

Using GPs, we can draw samples from a distribution to get plausible predictions of our underlying function f. We do this through the discretisation of our input space by using a large vector of function values $\mathbf{f}$ to represent our function [15]. This means that rather than predicting a label y_i for a given input $\mathbf{x}_i$, we instead model $\mathbf{f}$ directly as an infinite-dimensional Gaussian.

2.2.1 Gaussian Process Regression (GPR)

Like any Gaussian distribution, a GP is fully defined by its mean and covariance. For GPs, however, we generalise this further by adapting the concept of a mean function $m(x)$ and covariance function $k(x,x')$, also known as the kernel function [34]:

$$f \sim \mathcal{GP}(m,k) \tag{2.7}$$

Without any specific domain knowledge, $m(x)$ is often set to zero which leaves the kernel function to control the shape of the GP. More specifically, the kernel function describes how function values depend on each other and is used to compute an entry in the covariance matrix K for each input pair:

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) \tag{2.8}$$

To generate smooth functions, we expect strong positive correlations between function values of close input locations and weaker correlations for inputs locations far away. Furthermore, we also need to ensure that this kernel function is positive definite or else the GP will not define a valid Gaussian. A kernel which satisfies both of these requirements is the RBF kernel:

$$k(\mathbf{x}_i, \mathbf{x}_j) \;=\; \sigma_f^2 \exp\left(-\tfrac{1}{2} \sum_{d=1}^{D} (x_{i,d} - x_{j,d})^2 / \ell_d^2 \right) \tag{2.9}$$

where $x_{i,d}$ is the d-th component of the i-th input and σ_f^2, ℓ_d^2 are hyper parameters.

The kernel we choose specifies the set of potential functions we want to consider before observing data. This is known as the GP prior and represents our *a priori* beliefs about $\mathbf{f}$ [11]. In practice, we do not use $\mathbf{f}$ directly (as it is infinite-dimensional), but rather consider a finite subset of function values, denoted by $\mathbf{f}_*$, evaluated at some test locations $\mathbf{X}_*$ [15]. By the definition of a GP, we know that this subset is also normally distributed. Thus, we can write:

$$\mathbf{f}_* \sim \mathcal{N}(0, K(\mathbf{X}_*, \mathbf{X}_*)) \tag{2.10}$$

where $K(\mathbf{X}_*, \mathbf{X}_*)_{ij} = k(\mathbf{x}_{*,i}, \mathbf{x}_{*,j})$

Assuming our training labels $\mathbf{y}$ (observed at input locations $\mathbf{X}$) are generated from $\mathbf{f}$, we can add a noise variance to account for noisy observations in our observation model [11]:

$$y_i \sim \mathcal{N}(f_i, \sigma_y^2) \tag{2.11}$$

The joint distribution of $\mathbf{y}$ and $\mathbf{f}_*$ is then given by

$$p\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_*\end{bmatrix}\right) = \left(\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_*\end{bmatrix}; \mathbf{0}, \begin{bmatrix}K(X,X)+\sigma_y^2 & K(X,X_*)\\ K(X_*,X) & K(X_*,X_*)\end{bmatrix}\right) \tag{2.12}$$

From this, we can derive the posterior predictive distribution $p(\mathbf{f}_*|\mathbf{y})$ which is:

$$\mathcal{N}(K(X_*,X)(K(X,X)+\sigma_y^2)^{-1}\mathbf{y}, K(X_*,X_*)-K(X_*,X)(K(X,X)+\sigma_y^2)^{-1}K(X,X_*))$$

In the case of regression, we are interested in the maximum a posteriori (MAP) estimate which is the posterior mean of this distribution. If we are only estimating a particular point, we can use marginalisation to extract its mean (value) and standard deviation (uncertainty) [11].

2.2.2 Multi-task Gaussian Process

Multi-task Gaussian Processes (MTGPs) are an extension of the GP framework, designed to model multiple correlated tasks or outputs simultaneously. The key idea in MTGP is to extend the GP covariance function to incorporate the correlations between different tasks. This is often achieved using a multitask kernel, which is a combination of a standard kernel and a task-specific kernel that models the correlations between the tasks [7].

2.2.2.1 Multitask Kernel Functions

To model MTGPs, we define a multitask kernel function, which extends the covariance structure to consider both input and task similarities. Given two inputs $\mathbf{x}_i$, $\mathbf{x}_j$ and their respective tasks a, b, we can define the multitask kernel as [7]:

$$k_{MT}((\mathbf{x}_i,a),(\mathbf{x}_j,b)) = k(\mathbf{x}_i,\mathbf{x}_j)\times k_T(a,b) \tag{2.13}$$

Here, $k(\mathbf{x}_i,\mathbf{x}_j)$ is the input kernel (e.g., the RBF kernel in Equation (2.9)), and $k_T(a,b)$ is the task kernel that quantifies the similarities between tasks a and b. Like the input kernel, the task kernel should be positive definite to maintain a valid Gaussian Process.

2.2.2.2 Multitask Gaussian Process Regression

Assuming we have T tasks and N observations for each task, we stack the observations of all tasks into a single vector $\mathbf{y}_{MT} \in \mathbb{R}^{NT}$. We also stack the inputs and associate each with their respective tasks, creating a new input set $\mathcal{X}_{MT} = \{(\mathbf{x}_i,t_i)\}$ for $i = 1,\ldots,NT$, where t_i is the task index for the i-th input.

Following the notation in Section 2.2.1, we can then define the MTGP prior over the function values $\mathbf{f}_{MT_*}$ at test locations $\mathbf{X}_{MT_*}$ as:

$$\mathbf{f}_{MT_*} \sim \mathcal{N}(0, K_{MT}(\mathbf{X}_{MT_*},\mathbf{X}_{MT_*})) \tag{2.14}$$

where $K_{MT}(\mathbf{X}_{MT_*}, \mathbf{X}_{MT_*})_{ij} = k_{MT}((\mathbf{x}_{*,i}, t_{*,i}), (\mathbf{x}_{*,j}, t_{*,j}))$.

We can then extend the joint distribution of the observed outputs $\mathbf{y}_{MT}$ and test function values $\mathbf{f}_{MT_*}$, as shown in Equation (2.12), to include the multitask kernel structure:

$$p\left(\begin{bmatrix} \mathbf{y}_{MT} \\ \mathbf{f}_{MT_*} \end{bmatrix}\right) = \left(\begin{bmatrix} \mathbf{y}_{MT} \\ \mathbf{f}_{MT_*} \end{bmatrix}; \mathbf{0}, \begin{bmatrix} K_{MT}(\mathcal{X}_{MT}, \mathcal{X}_{MT}) + \Sigma_y & K_{MT}(\mathcal{X}_{MT}, \mathbf{X}_{MT_*}) \\ K_{MT}(\mathbf{X}_{MT_*}, \mathcal{X}_{MT}) & K_{MT}(\mathbf{X}_{MT_*}, \mathbf{X}_{MT_*}) \end{bmatrix}\right) \tag{2.15}$$

Here, Σ_y is a block-diagonal matrix where the t-th block contains the task-specific noise variances for the t-th task.

Using the joint distribution in Equation (2.15), we can derive the posterior predictive distribution $p(\mathbf{f}_{MT_*}|\mathbf{y}_{MT})$ for the multitask setting. The posterior mean of this distribution is given by:

$$\mu_{MT_*} = K_{MT}(\mathbf{X}_{MT_*}, \mathcal{X}_{MT})(K_{MT}(\mathcal{X}_{MT}, \mathcal{X}_{MT}) + \Sigma_y)^{-1}\mathbf{y}_{MT} \tag{2.16}$$

and the posterior covariance of this distribution is given by:

$$\Sigma_{MT_*} = K_{MT}(\mathbf{X}_{MT_*}, \mathbf{X}_{MT_*}) - K_{MT}(\mathbf{X}_{MT_*}, \mathcal{X}_{MT})(K_{MT}(\mathcal{X}_{MT}, \mathcal{X}_{MT}) + \Sigma_y)^{-1}K_{MT}(\mathcal{X}_{MT}, \mathbf{X}_{MT_*}) \tag{2.17}$$

Like before, we can use marginalisation to extract the mean and standard deviation for the desired task [7].

2.3 Copula-GP

In this section, we explore the semi-parametric approach behind Copula-GP (Kudryashova et al.) [22], which uses non-parametric marginals and parametric copulas to model high-dimensional dependencies. The overall framework is based on the inference for margins (IFM) training scheme [18] and can be summarised into two main stages:

1. In the first stage, univariate marginals (eCDF) are estimated using the non-parametric algorithm fastKDE [32], allowing data to be mapped onto the unit hypercube via the probability integral transform.
2. In the second stage, a copula mixture model is constructed by inferring the copula parameters from the empirical conditional copula [22].

Linear Copula Mixtures. The copula mixture model in Copula-GP is built using four bivariate copula families: Gaussian, Frank, Clayton, and Gumbel, see Figure 2.1. Each of these families has a single parameter that corresponds to the rank correlation. To capture upper tail dependencies and negative correlation, the model also includes rotated variants (at 90°, 180°, and 270°) of the Clayton and Gumbel copulas. Since the Independence copula is a special case in all of the copula families, it is added as a separate family with zero parameters.

Gaussian Processes. To parametrise each copula in the linear mixture model, the framework uses independent latent Gaussian processes with specific GPLink functions that map the GP variable onto the copula parameter domain. Moreover, GPs are also

used to parametrise concentrations $\varphi j(x)$, which are defined on a simplex (sum to one). The entire mixture model with M copula elements is parameterised by $2M-1$ independent GPs and requires $2M-1$ hyperparameters.

Approximate Inference. Due to the presence of latent variables with GP priors and an intractable posterior distribution, stochastic variational inference (SVI) is used as an approximate inference method in Copula-GP. The SVI implementation has a single evidence lower bound, allowing for joint optimisation of the GP hyperparameters and the parameters of the variational distribution. Training is carried out using the Adam optimiser with two learning rates, one for the GP hyperparameters and another for the variational distribution parameters. To avoid overfitting, a normal prior $\lambda \sim \mathcal{N}(0.5,1)$ is placed on the RBF kernel lengthscale parameter.

Bayesian Model Selection. For model selection, the framework uses the Watanabe-Akaike information criterion (WAIC), which is a fully Bayesian approach to estimating the AIC. The WAIC can be calculated from two measures: the log pointwise predictive density (lppd) and the effective number of parameters (p_{WAIC}). The lppd is a measure of the model's goodness of fit, while the p_{WAIC} represents the model's complexity:

$$\hat{\text{lppd}} = \sum_{i=1}^{N} \log\left(\frac{1}{S}\sum_{s=1}^{S} p(y_i|\theta_s)\right), p_{\text{WAIC}} = \sum_{i=1}^{N} V_{s=1}^{S} \log p(y_i|\theta_s)$$

where $\{\theta^s\}_S$ is a draw from a posterior distribution and $V_{s=1}^{S}$ represents the sample variance.

Using these terms, the WAIC can be computed as $\hat{\text{lppd}} - p_{\text{WAIC}} = -N \times \text{WAIC}$. In the model selection process, the goal is to choose the copula mixture that yields the lowest WAIC. To do so, Copula-GP offers two algorithms to create close-to-optimal copula mixtures: greedy and heuristic. The greedy algorithm is universal and can be used with any other copula families without adjustment, while the heuristic algorithm is fine-tuned to the specific copula families used in Copula-GP. For independent variables, both model selection algorithms will prefer the Independence model over other models, as it has the smallest number of parameters.

Copula vine constructions. In constructing high-dimensional copulas, Copula-GP employs the C-vine structure introduced in Section 2.1.2.2. While C-vines have the benefit of being suitable for neuronal data [26], one limitation of this structure is that the number of parameters grows exponentially with the number of variables. This, in turn, can lead to issues such as overfitting, high computational costs, and difficulties in model interpretation. To resolve this, Copula-GP incorporates a pruning mechanism for the C-vine construction that simplifies the resulting model without significantly compromising its accuracy. The pruning method involves identifying vine elements that contribute minimally to the model's overall performance. This is achieved by computing the difference in the WAIC when a particular vine element is removed. If the difference is below a predefined threshold, the element is considered insignificant and removed from the model. This process is iteratively applied, resulting in a simplified C-vine structure.

Goodness-of-fit. In Copula-GP, the accuracy of density estimation can be evaluated using the proportion of variance explained (R^2). This is calculated by comparing the estimated conditional cumulative distribution function $\text{ccdf}(u_2|u_1 = y)$ with the empirical conditional cumulative distribution function $\text{ecdf}(u_2|u_1 = y)$, using the following formula:

$$R^2(y) = 1 - \sum \frac{[\text{ecdf}(u_2|u_1 = y) - \text{ccdf}(u_2|u_1 = y)]^2}{[\text{ecdf}(u_2|u_1 = y) - u_2]^2} \tag{2.18}$$

Here, $R^2(y)$ quantifies the proportion of total variance of u_2 that can be explained by the copula model, given $u_1 = y$ and $u_2 = F(y_2) = 0.5$.

Mutual Information Estimation. In addition to constructing copula models conditioned on a task-related variable, Copula-GP can also estimate how much knowing one variable x contributes to knowing another variable $\mathbf{y}$ using mutual information. The mutual information $I(x, \mathbf{u})$, where $\mathbf{u} = \mathbf{F}(\mathbf{y})$, can be quantified through the unconditional and conditional entropies of the copula model:

$$I(y, \mathbf{u}) = H(\mathbf{u}) - \int H(\mathbf{u}|x = t)p(t)dt \tag{2.19}$$

Copula-GP offers two methods for estimating mutual information: the 'integrated' approach and the 'estimated' approach. The integrated approach assumes no conditional dependence in the marginals and uses direct integration of Equation 2.19. The estimated approach, on the other hand, does not impose any limitations on the conditional dependencies in the marginals, but requires an additional copula model for the unconditional distribution $p(\mathbf{y})$, in addition to a conditional copula model of $p(\mathbf{y}|x)$.

2.4 Related Work

2.4.1 Neural Copula Models

The use of copulas to model dependencies within neuronal populations was first proposed in "The Shape of Neural Dependency" by Jenison and Reale [17]. In this paper, Jenison and Reale discuss the limits of using the conventional product-moment correlation coefficient to model linear dependence on non-elliptical distributions. Instead, they present an alternative copula method which allows for the formation of "flexible multivariate distribution", eliminating "the implied reliance on the multivariate Gaussian" [17]. To this end, the authors suggest using the Akaike information criterion (AIC) to select the best-fitting copula for a given dataset. This method was used in [17] to model the joint conditional probability density of first-spike latency in two cortical neurons with the Ali-Mikhail-Haq (AMH) Copula.

Subsequent work by Onken et al. [29] expanded on this approach by exploring a broader range of parametric copula families to model neural dependence. They applied various copula-based models, including Clayton, Gumbel-Hougaard, Frank, Ali-Mikhail-Haq, and multi-parameter FGM copulas, alongside the discretised multivariate normal (MVN) distribution to analyse spike data from the prefrontal cortex of an awake behaving macaque. Their results revealed that the Clayton copula family provided the best fit for

their data and that the likelihood for the copula-based models was significantly larger than for the discrete MVN model. Additionally, Onken et al. also investigated the presence and importance of higher-order interactions by comparing different numbers of parameters for the FGM copula. They discovered that the FGM model with third-order interactions offered a significantly better fit for the data than the model with only pairwise interactions, suggesting that higher-order interactions played a crucial role in terms of model fit and information content. Nevertheless, they were unable to generalise this method to population sizes with more than 20 neurons as the proposed method became too computationally demanding for a higher number of neurons.

Continuing along the same vein, Onken et al. [30] employed mixtures of copulas and the so-called flashlight transformation to provide a more flexible approach for investigating dependencies in neuronal populations. This technique enabled the tail dependence of an arbitrary copula to be shifted into any orthant of the joint distribution, which proved particularly useful for modelling distributions of spike counts. By applying this technique to data from the macaque prefrontal cortex, it was shown that copula-based distributions with negative binomial margins were better suited for modelling spike-count distributions than the commonly-used multivariate Poisson latent variables distribution and the multivariate normal.

Additional work studying the cortex of macaque monkeys with copula models has also been undertaken by Berkes et al. [6]. In their study, Berkes et al. demonstrated how to construct non-independent joint distributions over firing rates using copulas. Furthermore, they also derived a Maximum Likelihood (ML) procedure for estimating parameter values for their neural copula model. This method was tested on neural data recorded from the pre-motor cortex while a macaque executed a centre-out reaching task. Results showed that the dependency structure between 54% of neuron pairs could be represented by the Gaussian-like dependency of the Frank copula. More interestingly, a significant portion of neuron pairs revealed tail dependencies, where 38% of them were either concentrated in the upper tails (Gumbel copula) or the lower tails (Clayton copula).

In [6], Berkes et al. identified an important challenge of generalising the aforementioned copula methods to distributions over responses of n neurons rather than just neuron pairs. Most copula families are bivariate and, as such, are unlikely to capture the full complexity of non-homogeneous dependency structures within neuronal populations. One exception to this limitation is the Gaussian copula, which may be extended to the n-variate case. However, Berkes et al. argue that although it is mathematically feasible, using the Gaussian copula becomes ”prohibitively expensive to fit in high dimensions” [6].

Work by Onken and Panzeri [31] and Mitskopoulos et al. [26] address this dimensionality problem by incorporating vine copulas into their frameworks. Both [31] and [26] use the C-vine; however, the former employs a parametric approach, whereas the latter uses a non-parametric method with flow-based vine copulas.

Other non-parametric approaches to neural copula modelling have also been proposed in work by Sacerdote et al. [35]. Instead of inferring an underlying copula parameter describing the dependence structure, this non-parametric method involves analysing

copula scatter plots to interpret relationships between spike trains and using association indexes like Kendall's tau to quantify the strength of these relationships. Non-parametric approaches like [35] and [26] offer the advantage of avoiding the rigid constraints imposed by assuming an underlying parametric distribution. However, such methods tend to be less robust to data sparsity, which is often present in neuronal datasets where the number of trials is limited but the dimensionality is high. In such scenarios, assumptions about the dependencies between variables can be helpful, allowing parametric models to offer a degree of specificity and interpretability that non-parametric approaches may lack.

2.4.2 Gaussian Process Copula Models

Although work combining copulas and GPs in the field of neuroscience is limited, this has previously been applied to analyse other types of data. In [25], Lopez-Paz et al. proposed GPVINE, a method for scaling copulas to high-dimensional data by constructing a hierarchy of conditional bivariate copulas using vine factorisations. Rather than assuming that a copula is independent of its conditioning variables (something which is often done to simplify inference), Lopez-Paz et al. used Gaussian Process Regression to learn the latent functions describing these dependencies. Following this approach, it was found that the resulting predictive posterior distribution did not have an analytical solution but could be approximated using Expectation Propagation (EP). To evaluate this method, GPVINE was compared against a baseline model using the simplifying assumption (SVINE) and a state-of-the-art alternative based on maximum local-likelihood methods (MLLVINE). Evaluation was performed on various datasets, ranging from weather recordings to mineral concentration levels. Results showed that GPVINE consistently outperformed SVINE while also performing favourably with respect to MLLVINE.

Work by Hernández-Lobato et al. [13] later extended the conditional copula model of [25] by accommodating copulas with multiple parameters for bivariate data. This was previously computationally infeasible in [25], as it required numerical calculations of multidimensional integrals within the EP inference. As a solution, Hernández-Lobato et al. proposed Gaussian process conditional models (GPCC), which allowed the inclusion of copulas such as Student's t and asymmetric copulas. This extension was shown to be beneficial in improving the model's performance when evaluated on synthetic data and financial time series.

Chapter 3

Methods

This chapter details the methodology for incorporating new copula families into Copula-GP. We begin in Section 3.0.1 with a brief overview of the essential steps required to implement a new copula family within the framework. We then detail how this process is executed for each of the copula families we add. Finally, in Section 3.0.5, we propose a modified architecture for Copula-GP, which allows for the inclusion of copulas with multiple parameters.

3.0.1 General Approach

The Copula-GP package is based on the PyTorch library [33], which we use to implement the following components:

Conditional Copula: Copula-GP uses conditional copulas to model the dynamic dependence structure between neurons or behavioural variables, given a continuous task-related variable. To introduce new copula families into the framework, we must derive their conditional form using the approach described in Section 2.1.1.2.

Inverse Conditional Copula: In addition to the conditional copula, we also need to obtain its inverse. This is necessary to enable sampling from each copula family using the conditional distribution method outlined in Section 2.1.1.2.

Copula Density: The copula density is used to compute the log-likelihood term during training with stochastic variational inference. The copula density can be obtained by differentiating the copula function with respect to each variable.

GPLink Function: Most of the copula families used in Copula-GP have restricted parameter domains that do not coincide with the range of the GP variable (with the exception of the Frank copula). For instance, the parameter of the Gaussian copula ranges from -1 to 1, but the GP variable $f \in \mathbb{R}$ is on the real line. Thus, we need a GPLink function that maps from $\mathbb{R}$ to the parameter domain: $\theta_j = \text{GPlink}_{c_j}(f_j)$, $\mathbb{R} \to \text{dom}(c_j)$.

3.0.2 The Ali-Mikhail-Haq Copula

Following this approach, the first copula we add is the Ali-Mikhail-Haq (AMH) copula, which belongs to the Archimedean copula class [3]. The AMH copula is given by the following function:

$$C(u,v) = \frac{uv}{1+\theta(1-u)(1-v)} \tag{3.1}$$

where θ is the copula parameter that lies within the range $[-1,1)$. Like the Gaussian copula, the AMH copula reduces to the Independence copula when $\theta = 0$.

The AMH copula can model both tail dependence and tail independence, depending on the value of θ. In particular, when $\theta = 1$, the AMH copula exhibits left tail dependence [23]. This tail dependence is generally weak, so the AMH copula is best suited for modelling modest dependencies [16].

In this project, we decided to add the AMH copula into the Copula-GP framework for three main reasons:

1. **Diversity**: The AMH copula is particularly well-suited to model data that is close to independent around central values but exhibits weak tail dependence around extreme values. This is in contrast to current copula families in Copula-GP, which either capture no tail dependence or strong tail dependence.
2. **Relevance**: The AMH copula has been successfully applied in previous research to model dependencies between cortical neurons [17]. Given that the primary application for Copula-GP is in neuroscience, incorporating the AMH copula is a logical choice.
3. **Tractability** The AMH copula belongs to the Archimedean copula class, which has numerous useful properties, including an explicit functional form.

3.0.2.1 The Conditional AMH Copula

As our first step, we need to obtain the conditional form of the AMH copula. To do this, we take the partial derivative of the copula function with respect to the conditioning variable U.

$$C(v|u) = \frac{\partial C(u,v)}{\partial u} = \frac{v(1+\theta(1-v))}{(1+\theta(1-u)(1-v))^2} \tag{3.2}$$

The resulting equation cannot be directly implemented, as it will cause numerical issues for certain combinations of values. For instance, a division by zero error may occur when $u \to 0$, $v \to 0$, and $\theta \to 1$. To resolve this, we limit the range of the variables with a small $\varepsilon = 10^{-7}$ such that $u,v \in [\varepsilon, 1]$ and $\theta \in [-1, 1-\varepsilon]$. The value of 10^{-7} is determined experimentally by analysing at which ε `NaN` values occur.

Another challenge when working with numbers between 0 and 1 is that they are prone to numerical underflow when multiplied together. To further increase the numerical stability of our implementation, we employ the log-sum-exp trick commonly used in

probabilistic modelling to the conditional copula: $C(u|v) = \exp(\log(v(1+\theta(1-v))) - 2\log(1+\theta(1-u)(1-v)))$.

3.0.2.2 The Inverse Conditional AMH Copula

Once we have the conditional form of the AMH copula, we can algebraically derive the inverse conditional CDF by setting an auxiliary variable t equal to $C(v|u)$ and solving for v. After some algebraic grunt work, we get the following expression for the inverse $C^{-1}(t|u)$:

$$\frac{1-\theta-2t\theta+2t\theta^2+2t\theta u-4t\theta^2u+2t\theta^2u^2 \pm \sqrt{1-2\theta+\theta^2+4t\theta u-4t\theta^2u+4t\theta^2u^2}}{2(-\theta+t\theta^2-2t\theta^2u+t\theta^2u^2)} \tag{3.3}$$

We discard the option in which the square root yields a positive value as that brings v outside of the $[0,1]$ range. Furthermore, we also log-transform the inverse to increase numerical stability.

We observe that as $\theta \to 0$, both the numerator and the denominator of Equation 3.3 tend to zero. To avoid this issue, we leverage the fact that the AMH copula reduces to the Independence copula when $\theta = 0$. In other words, $C(v|u)$ and $C^{-1}(t|u)$ can be approximated with v and t respectively when θ is sufficiently small. We implement this by setting a threshold λ and define a modified inverse $C_*^{-1}(t|u)$:

$$C_*^{-1}(t|u) \begin{cases} t & \text{if } |\theta| < \lambda \\ C^{-1}(t|u) & \text{otherwise} \end{cases} \tag{3.4}$$

Likewise, we update $C_*(v|u)$ to return v whenever $|\theta| < \lambda$. To determine the value of λ, we run unit tests by transforming randomly generated samples twice, first using $C(v|u)$ and then using $C^{-1}(t|u)$. We perform these tests for 500 values of θ linearly spaced between -1 and 1. At $\lambda = 0.005$, the transformed samples differ from the original samples by less than the distance tolerance of 10^{-4} set in the original Copula-GP package.

3.0.2.3 The AMH Copula Density

As discussed in previous sections, the copula density can be obtained by differentiating the copula function with respect to each variable. Since we already have the conditional form of the AMH copula, we can derive the density by taking the partial derivative with respect to v:

$$c(u,v) = \frac{(1-2\theta v)(1+\theta(1-u)(1-v))^2 - 2\theta v(1-u)(1+\theta(1-v))(1+\theta(1-u)(1-v))}{(1+\theta(1-u)(1-v))^4} \tag{3.5}$$

As before, we log-transform this expression to increase numerical stability.

3.0.2.4 Formulating the AMH GPLink Function

For the GPLink function of the AMH family, we need to define a function which maps from $\mathbb{R}$ to [-1,1]. Conveniently, the AMH copula has the same parameter range as the Gaussian copula, which means that we can use its GPLink function: $\text{Erf}(f/1.4)$, for the AMH family as well. Here, Erf represents the error function [37].

3.0.3 The Joe Copula

Next, we move on to the Joe copula which is another copula belonging to the Archimedean class. Like the Gumbel copula, it has a strong clustering of values in the upper right orthant but exhibits a thicker tail, see Figure 2.1. The explicit form of the Joe copula is given by [19]:

$$C(u,v) = 1 - [(1-u)^\theta + (1-v)^\theta - (1-u)^\theta(1-v)^\theta]^{\frac{1}{\theta}} \tag{3.6}$$

The parameter θ takes values in the range $[1,\infty]$, and when $\theta = 1$, the Joe copula reduces to the Independence copula. As θ increases, the tail dependence becomes stronger, allowing the Joe copula to model more extreme joint behaviours.

We chose to incorporate the Joe copula into Copula-GP for the following reasons:

1. **Relevance** Previous work such as [29] and [22] have shown that spike data usually exhibit heavy-tailed dependence, which can be captured by the Joe copula's upper tail dependence.
2. **Tractability** The Joe copula is an Archimedean copula; thus, it has numerous nice properties, including an explicit functional form.

3.0.3.1 The Conditional Joe Copula

As before, we take the partial derivative with respect to the conditioning variable u to derive the conditional form of the Joe copula:

$$\frac{\partial C(u,v)}{\partial u} = (1-u)^{\theta-1}(1-(1-v)^\theta)[(1-u)^\theta + (1-v)^\theta - (1-u)^\theta(1-v)^\theta]^{\frac{1}{\theta}-1} \tag{3.7}$$

Given that θ can range from 1 to $+\infty$ and $u,v \in [0,1]$, we apply the log-sum-exp trick once more to prevent numerical underflow. Furthermore, we also restrict the range of u,v to be within $[\varepsilon, 1-\varepsilon]$ to avoid any division by zero errors.

3.0.3.2 The Inverse Conditional Joe Copula

Unlike the AMH copula, the conditional form of the Joe copula is not invertible. This means that we have to rely on numerical methods to calculate the inverse. Specifically, we are interested in the set of root-finding algorithms which can solve the equation $0 = t - C(v|u)$ for v.

To simplify our problem, we reparametrise our variables and let $x = (1-u)^\theta$, $y = (1-v)^\theta$. This allows us to solve for y instead v which will provide a much more

numerically stable solution. Moreover, we also take the logarithm to prevent any vanishing values. The function which we want to find the root of is then:

$$0 = \log(t) - (\log(x) - \log(1-u) + \log(1-y) + (\frac{1}{\theta} - 1)\log(x+y-xy) \tag{3.8}$$

There are several root-finding algorithms capable of solving the equation above. In this project, we consider the class of Householder's methods which use derivatives to iteratively approximate the root [39]. Although convergence is not always guaranteed, Householder's methods typically exhibit faster convergence rates compared to more numerically stable techniques, such as the Bisection Method [20]. We motivate this trade-off between numerical stability and convergence speed by the need for efficiency in a machine-learning framework like Copula-GP.

Out of the Householder's methods, we evaluate two options: the Newton-Raphson's method and the Halley's method, which each correspond to the first and second-order variants of this class. The Newton-Raphson method employs the first derivative, while Halley's method utilises both the first and second derivatives. At each iteration, we update our guess as follows [20]:

$$y_{n+1}^{(NR)} = y_n - \frac{f(y_n)}{f'(y_n)} \tag{3.9}$$

$$y_{n+1}^{(H)} = y_n - \frac{2f(y_n)f'(y_n)}{2[f'(y_n)]^2 - f(y_n)f''(y_n)} \tag{3.10}$$

Although Halley's method generally converges faster, it comes at the cost of added complexity and computational overhead. This, in turn, might offset the additional speed benefits gained in faster convergence. To identify the most efficient option, we implement both methods and compare their run times through experiments. We detail the procedure and present our results in Appendix A.

3.0.3.3 The Joe Copula Density

Using the expression for the conditional Joe copula, we can differentiate with respect to v to obtain the copula density. To keep the final expression succinct, we continue with our previous notation where $x = (1-u)^\theta$ and $y = (1-v)^\theta$.

$$c(u,v) = \frac{[x+y-xy]^{\frac{1}{\theta}-2}x^{1-\frac{1}{\theta}}y^{1-\frac{1}{\theta}}[(\theta-1)+x+y]}{u^2v^2} \tag{3.11}$$

3.0.3.4 Formulating the Joe GPLink Function

To use the Joe copula in Copula-GP, we need a GPLink function that maps from $\mathbb{R}$ to the parameter domain of $[1,\infty]$. Fortunately, we can reuse the GPLink function of the Gumbel copula: $1+\exp(0.1\cdot f)$, as it has the same parameter range.

3.0.4 The Student's t Copula

Our final addition to the new set copula families is the Student's t copula which we write as t-copula for short. The t-copula is derived from the multivariate Student's t distribution and is an elliptical copula that allows for both upper and lower tail dependence, see Figure 2.1. The Student's t copula can be defined as [19]:

$$C(u,v) = T_\nu(T_\nu^{-1}(u), T_\nu^{-1}(v); \rho) \tag{3.12}$$

where T_ν and T_ν^{-1} denote the CDF of the multivariate Student's t distribution and its inverse (PPF), respectively.

Unlike the other copula families introduced in this report, the t-copula is controlled by two parameters. The first parameter, ν, signifies the degrees of freedom and lies in the range $[2,\infty)$. Meanwhile, the second parameter, ρ, represents the correlation coefficient and spans the interval $[-1,1]$. As ν grows increasingly larger, the t-copula converges to the Gaussian copula, and when $\rho = 0$, the t-copula becomes the Independence copula. Lower values of ν lead to heavier tails and, consequently, stronger tail dependence.

To use the t-copula in the Copula-GP framework, we need to modify the original package as it currently only accommodates copulas with a single parameter. Since this modification is rather extensive, we describe this separately in Section 3.0.5.

Our motivation for adding the t-copula to Copula-GP is threefold:

- **Relevance.** The t-copula is a suitable choice for many real-world problems and has previously been useful in Gaussian Process copula models to analyse financial time series data [13].
- **Flexibility.** The Gaussian copula has been extensively used for modeling various dependence structures; however, it falls short in cases where extreme events exhibit stronger correlations. The t-copula offers more flexibility due to its additional parameter, ν, which enables the simulation of Gaussian-like dependence patterns while incorporating tail dependence.
- **Novelty.** The t-copula has two parameters and serves as an excellent starting point for expanding the Copula-GP framework to incorporate copulas with multiple parameters.

3.0.4.1 The Conditional Student's t Copula

Since the t-copula does not have an analytical form, we use the bivariate conditional Student's t distribution to obtain the conditional copula instead. We use results from Ding [9] to guide our derivation. In this paper, Ding shows that the multivariate conditional t-distribution for two variables $X_1 \sim t_{p_1}(\mu_1, \Sigma_{11}, \nu)$ and $X_2 \sim t_{p_2}(\mu_2, \Sigma_{22}, \nu)$ with dimensionality p_1 and p_2 respectively can be expressed as:

$$X_2 \mid X_1 \sim t_{p_2}\left(\mu_{2|1}, \frac{\nu + d_1}{\nu + p_1}\Sigma_{22|1}, \nu + p_1\right) \tag{3.13}$$

where $\mu_2|1 = \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(X_1 - \mu_1)$ is the conditional mean, $d_1 = (X_1 - \mu_1)^T\Sigma_{11}^{-1}(X_1 - \mu_1)$ is the squared Mahalanobis distance, and $\Sigma_{22|1} = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}$ is the conditional scale matrix.

In the bivariate case where both $p_1 = 1$ and $p_2 = 1$, this translates to:

$$x_2 \mid x_1 \sim t_1\left(\rho x_1, \frac{\nu + x_1^2}{\nu+1}(1-\rho^2), \nu+1\right) \tag{3.14}$$

Now let $u = T_\nu(x_1)$ and $v = T_\nu(x_2)$. The conditional t-copula is then:

$$C(v|u) = T_{\nu+1}\left(\frac{T_\nu^{-1}(v) - \rho T_\nu^{-1}(u)}{\sqrt{\frac{\nu+T_\nu^{-1}(u)^2}{\nu+1}(1-\rho^2)}}\right) \tag{3.15}$$

3.0.4.2 The Inverse Conditional Student's t Copula

To find the inverse conditional t-copula, we set an auxiliary variable $t \in [0,1]$ equal to the left-hand side of Equation 3.15 and solve for v. This yields:

$$C^{-1}(t|u) = T_\nu\left(T_{\nu+1}^{-1}(t)\sqrt{\frac{\nu + T_\nu^{-1}(u)^2}{\nu+1}(1-\rho^2)} + \rho T_\nu^{-1}(u)\right) \tag{3.16}$$

3.0.4.3 The Student's t Copula Density

The t-copula density is given by [1]:

$$c(u,v) = \frac{\Gamma(0.5\nu)\Gamma(0.5\nu+1)(1+\frac{T_\nu^{-2}(u)+T_\nu^{-2}(v)-2\rho T_\nu^{-1}(u)T_\nu^{-1}(v)}{\nu(1-\rho^2)})^{-\frac{\nu+2}{2}}}{\sqrt{1-\rho^2}\Gamma(0.5(\nu+1))^2(1+\frac{T_\nu^{-2}(u)}{\nu})^{-\frac{\nu+1}{2}}(1+\frac{T_\nu^{-2}(v)}{\nu})^{-\frac{\nu+1}{2}}}$$

where Γ denotes the gamma function [38].

3.0.4.4 Formulating the GPLink Functions for the Student's t Copula

As the t-copula has two parameters, we must specify two GPLink functions - one for the correlation coefficient (ρ) and another for the degrees of freedom (ν). For the correlation coeffcient, we can reuse the GPLink function of the Gaussian copula: $\text{Erf}(f/1.4)$, since it has the same range of -1 to 1. However, because none of the other copula families possess the parameter domain of $[2,\infty)$ for ν, we must devise a new GPLink function.

We draw inspiration from the GPLink functions of the Clayton ($\theta \in [0,\infty]$) and Gumbel ($\theta \in [1,\infty]$) families, which are $\exp(0.2 \cdot f)$ and $1+\exp(0.1 \cdot f)$, respectively. Note that the coefficients 0.2 and 0.1 are hyper-parameters used to regulate the effective learning rate of each copula parameter. These coefficients are necessary for model selection as they ensure that different copula families converge at similar rates despite the non-linear nature of the GPLink functions. Thus, we define our GPLink function for ν to be $2+\exp(\alpha \cdot f)$, where α is a hyper-parameter we tune experimentally. The results of our experiments are presented in Section 4.1.

3.0.4.5 Computing the Student's t CDF

A major challenge we encountered whilst implementing the t-copula was that the cumulative distribution function (CDF) of the Student's t-distribution T_ν and its inverse T_ν^{-1} were not supported by the PyTorch library. This was a considerable obstacle since both functions are requisites for constructing the t-copula, as per Equations 3.15, 3.16, and 3.0.4.3.

To address this issue, we considered two potential solutions. The first involved using a statistical library like Scipy [36] that has pre-built functions for computing the CDF and inverse CDF of the t-distribution. However, using Scipy would necessitate transferring the PyTorch tensor from the GPU to the CPU, which could lead to significantly longer training times. Instead, we opted to preserve performance by implementing the CDF and its inverse from scratch using PyTorch.

Student's t-distribution CDF. The t-distribution is related to the beta distribution in that if a random variable Y follows a t-distribution with ν degrees of freedom, then $X = \frac{1}{2} + \frac{1}{2}\frac{Y}{\sqrt{\nu+Y^2}}$ follows a beta distribution with the shape parameters $a = \frac{\nu}{2}$ and $b = \frac{\nu}{2}$. This relationship is often used to compute the CDF and PPF of the t-distribution [38], and is what we used in this project as well. Computing the CDF of the beta distribution is not trivial, however, as it is also not supported by PyTorch. Thus, to implement it in an efficient and numerically stable way, we based our code implementation on the Cephes Mathematical Functions Library [27], which was originally written in C. We modify it such that our implementation works with batches of values, specifically with PyTorch data structures (tensors) to allow for GPU acceleration.

Student's t-distribution PPF: To approximate the percentile point function of the t-distribution $T_\nu^{-1}(u) = x$, we use our implementation of the t-CDF in combination with the Newton-Raphson method described in Section 3.0.3.2. We chose this particular root-finding method for two reasons: (1) Although the t-CDF is not available in PyTorch, the t-PDF is. Thus, we have an efficient way to calculate the derivative of the equation $0 = u - T_\nu(x)$ with respect to x. (2) The Newton-Raphson method converges faster than simpler methods such as the Bisection method. This is especially important as running the forward computation of $T_\nu(x)$ is computationally demanding.

Using the Newton-Raphson method, we iteratively update our guess for x as per Equation 3.9. However, we use an additional trick to improve the convergence speed of our method. Instead of initialising x randomly, we take advantage of the fact that the normal distribution is a good approximation of the t-distribution, especially for large values of ν. Since the PPF for the normal distribution is available in PyTorch, we set our initial guess of x to be the PPF of the normal distribution evaluated at u. This enables our method to converge in less than five iterations for a distance tolerance of 10^{-5}, effectively reducing the runtime by a factor of three.

We provide our code implementation of both the Student's CDF and PPF in Appendix B.

3.0.5 Extending for Multiple Parameters

This section proposes a modified architecture for the Copula-GP model which allows for the inclusion of copulas with arbitrarily many parameters. Note that as a simplifying assumption, we model each parameter of the copula independently.

Our approach builds upon the copula mixture structure from the original framework, where a multitask GP with $2M-1$ tasks is used to parameterize M copulas. In our modified version, we extend this to $\sum_{m=1}^{M} n_m + M - 1$, where n_m represents the number of parameters for the m-th copula. This is achieved by introducing an additional field (`param_no`) in each copula class to keep track of the number of parameters for a given family. If there are M copulas in a copula mixture, we initialise $\sum_{m=1}^{M} \texttt{param\_no} + M - 1$ independent tasks.

To implement each copula family within this architecture, we add an extra dimension to the tensor (M-way array) containing the copula parameters for each copula instance in the mixture. While this allows for multiple parameters to be stored together, it also results in a dimensionality mismatch between the GP output and the parameter tensors. We address this issue by effectively ravelling and unravelling the stack of copula parameters during each training iteration. Moreover, we stack the GPLink functions for each individual parameter to create a collection of GPLink functions which operate on all parameters simultaneously. Figure 3.1 depicts a schematic diagram of the structure.

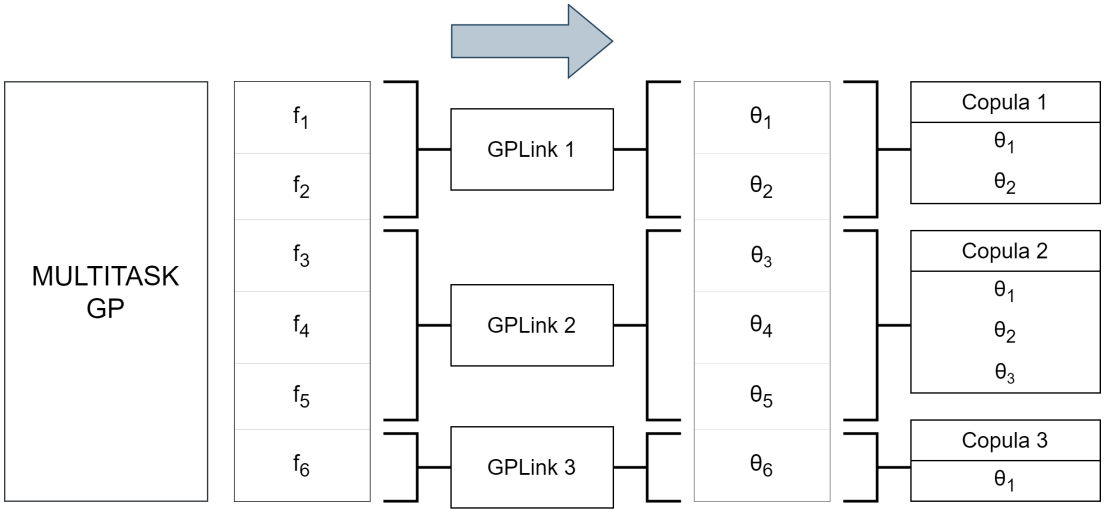

Figure 3.1: Modified architecture for Copula-GP which allows for copulas with multiple parameters to be used. Note that this figure only covers the copula parameters; the pipeline for the copula concentrations remains unchanged.

As shown in Figure 3.1, this extended framework not only accommodates copulas with any number of parameters but also mixtures composed of any number of copulas, each with any number of parameters.

Chapter 4

Results

This chapter details the experiments conducted in this project to evaluate and configure the implementation described in Chapter 3. The first experiment described in Section 4.1 aims to identify the optimal hyperparameter for the GPLink function of ν. The remainder of the experiments is concerned with evaluating the extended Copula-GP model against the original version and other state-of-the-art methods.

4.1 Tuning the GPLink Coefficient

In Section 3.0.4.4, we developed the GPLink function for the ν parameter of the t-copula to be $\exp(\alpha \cdot f) + 2$. The goal is to find the optimal value of α that ensures the t-copula converges at similar rates to other copula families while allowing the model to train effectively. To achieve this, we conduct two experiments. The first experiment is performed on the UCI shuttle dataset [8], which was originally used to fine-tune the GPLink hyperparameters. The second experiment is conducted on data generated using the copula library in R [14].

For the UCI shuttle dataset experiment, our objective is to study the convergence rates of different α values when the empirical copula density does not fit that of the t-copula. We do this by comparing the convergence rate of the GaussianGumbel180°mixture copula (which is the closest approximation of the t-copula in the original framework). Since neither of the two copulas fit the data, we expect both models to train poorly on the dataset. However, we are looking to examine which α configurations lead to similar losses and WAIC.

The learning curves for the t-copula on the shuttle dataset are depicted in Figure 4.1. Our experiments reveal that, in contrast to the GPLink functions for the Clayton and Gumbel families, using a small α coefficient (e.g., 0.1 or 0.2) to decelerate the convergence of the copula parameter results in significantly poorer learning. This finding can be ascribed to the unique nature of the t-copula's ν parameter compared to the copula parameters of the Clayton and Gumbel copulas. The t-copula's ν parameter dictates only the strength of tail dependence, whereas the Clayton and Gumbel copula parameters determine the overall dependence structure. As tail correlations are present only near extreme values,

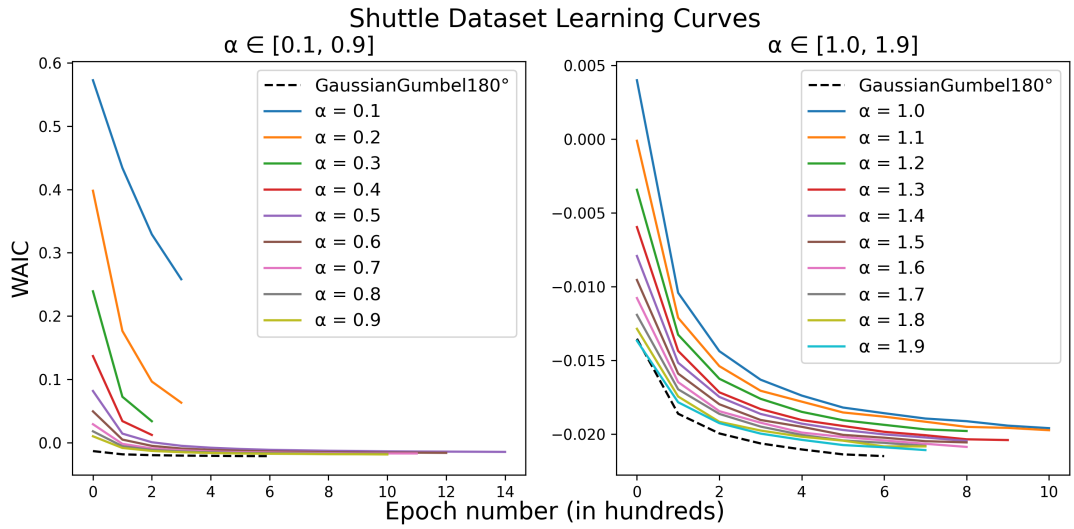

Figure 4.1: Learning curves of the t-copula for different values of α (GPLink hyperparameter). Left: $\alpha \in [0.1, 0.9]$. Right: $\alpha \in [1.0, 1.9]$.

the ν parameter operates on a different scale. For $\alpha > 0.4$, the learning curves for the t-copula behave similarly to that of the GaussianGumbel180°mixture copula, with some taking longer to converge than others.

After eliminating α values ≤ 0.4 based on the first experiment, we further filter our selection based on model performance by generating data from the t-copula using the copula package in R [14]. We vary the degrees of freedom but keep the correlation coefficient fixed. Our objective is not to compare losses with the GaussianGumbel180°copula, as we anticipate our model to train better. Instead, we aim to identify the α value that results in the most favourable training, which is indicated by the lowest WAIC.

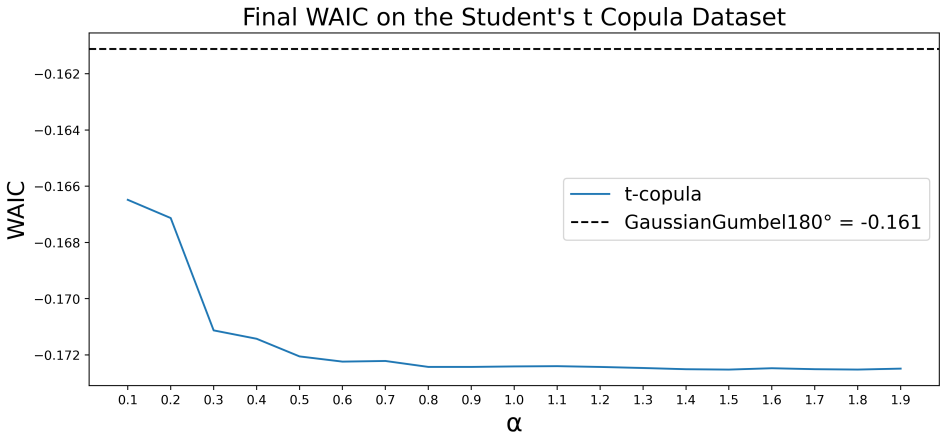

Figure 4.2: Final WAIC scores for different α values on the Student's t copula dataset.

Figure 4.2 shows the final WAIC of the t-copula model trained using different values of α. The results demonstrate that all α values provide better WAIC scores than the baseline copula mixture model. More importantly, the WAIC decreases rapidly from $\alpha = 0.1$ to $\alpha = 0.8$, after which it remains essentially unchanged. We thus opt for

the conservative choice of $\alpha = 1.0$ (neither slow down nor speed up) as our default value for the GPLink hyperparameter: $\nu = \exp(1.0 \cdot f) + 2$. Nonetheless, we leave our implementation of the `TCopula()` class to include the GPLink coefficient as an optional argument to allow for further analysis.

4.2 Model Evaluation

Our primary evaluation method involves comparing the original Copula-GP model with the extended model. The objective of these experiments is to assess the value of incorporating new copula families into the framework. Specifically, we aim to determine whether the new families provide unique and valuable dependence structures that cannot be easily replicated using copula mixtures from the original model. To this end, we use two evaluation metrics. The first metric is the WAIC introduced in Section 2.3, which considers both the predictive performance and complexity of the model. This metric is particularly relevant since Copula-GP uses the WAIC for Bayesian model selection and thus allows us to determine whether the added copula families will be preferred over other mixtures in the selection algorithm. The second metric we use to evaluate the goodness-of-fit is the coefficient of determination R^2 (see Section 2.3 for details). This allows us to measure the accuracy of the density estimation by comparing the empirical conditional CDF and the estimated conditional CDF.

In order to ensure the correctness of our implementation and prevent any affirmative bias, we generate data for each copula family using the copula [14] library in R. This allows us to conduct inference with known underlying parameter functions, which we use to verify the accuracy of our models and to identify their limitations. To model the copula parameter as it changes across time, we use transformed sinusoidal waves, as the shape of these functions can be well reproduced with the RBF kernel. Using this data, we compare the fit provided by the added copula families against the copula mixtures selected by the heuristics algorithm in the `select_copula` module of the Copula-GP package.

4.2.1 Evaluating the AMH Copula

We begin our evaluation with the AMH copula, which we describe in Section 3.0.2 as being well-suited for modelling data with weak dependence. To generate samples, we use the `amhCopula` class from the R copula package and set $\theta = 0.99\sin(2\pi x)$ for 9900 linearly spaced x-values between $[0,1]$. This allows us to examine the dependence structure of the AMH copula across its entire parameter domain. Note that since these samples will have uniform marginals by construction, we skip the marginal estimation step and perform inference directly using our implementation of the AMH copula. The inferred GP parameter and the true copula parameter are shown in Figure 4.3. (Note that the shaded areas of all subsequent GP plots represent function values within two standard deviations of the mean function). We find that Copula-GP is able to correctly infer the underlying function parameterising the AMH copula. The mean of the GP follows the trajectory of the sine waves closely, except at points of inflexions, where it slightly overestimates the slope of the curve. The GP has the lowest uncertainty when

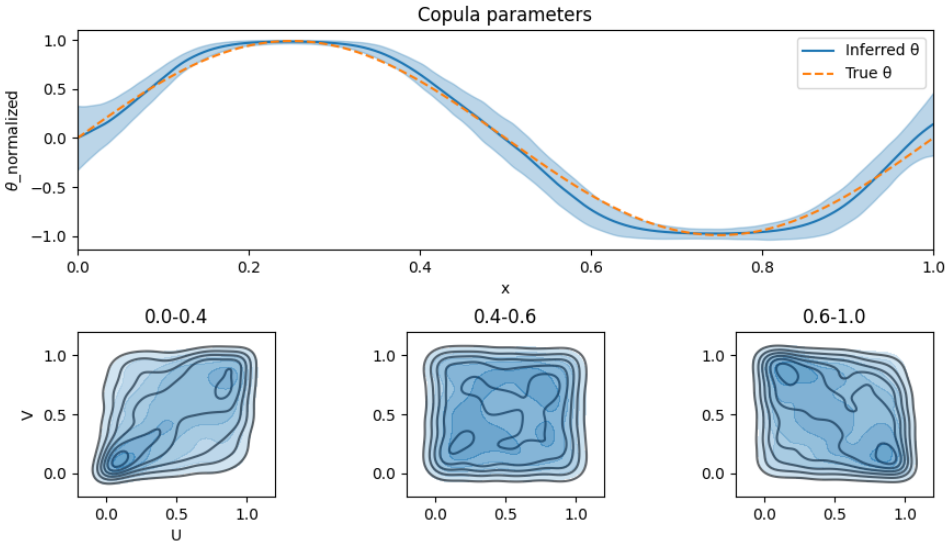

Figure 4.3: Top: Inferred GP parameter of the AMH copula in Copula-GP alongside the true parameter function used to generate the data. Bottom: The empirical copula density (black) and the estimated copula density (blue) during different intervals of x-values.

$\theta = \pm 0.99$, which is expected as the dependence structure is most defined around those points. Figure 4.3 also shows the empirical copula density alongside the inferred copula densities as the θ parameter oscillates between $[0.99, -0.99]$. The dependence is not particularly strong at either the maximum or minimum parameter values, but the AMH copula does exhibit tail correlations at $\theta = 0.99$, which is in line with previous work by Kumar [23]. Notably, this tail dependence does not occur when $\theta = -0.99$ (see density plot for $x \in [0.0, 0.4]$ and $x \in [0.6, 1.0]$).

Given that the AMH copula exhibits tail dependence, albeit weak, we hypothesise that a copula mixture modelling such dependence patterns would consist of either the Gumbel or Clayton families, as they are the only families in the original framework with tail correlations. Moreover, since these families are monotonic and cannot accommodate negative dependence, we anticipate that the Gaussian or Frank copula would also be included in the mixture to account for shifting signs of dependence. Finally, since the AMH copula's dependence structure is akin to that of the independence copula for central values on the unit square, we hypothesise that the independence copula would be present in the mixture, even when $\theta \neq 0$. The results from running the heuristics algorithm for selecting the optimal copula mixture are shown in Figure 4.4.

Using the original Copula-GP package, we find that the optimal copula mixture for modelling the AMH copula's dependence pattern consists of the Clayton (0°), Gaussian, and Independence copulas, as shown in Figure 4.4. This copula mixture is parameterised by 5 GPs in total. Interestingly, the GPs of the copulas' parameters and concentrations reveal the distinct roles each family has in reconstructing the dependence pattern of the AMH copula. For example, the Gaussian copula parameter displays a similar sinusoidal wave pattern as the true AMH parameter, albeit with lower amplitude, indicating that the Gaussian copula is responsible for modelling the "central dependence". Since the Gaussian copula does not exhibit tail dependence, but the AMH copula does, the copula

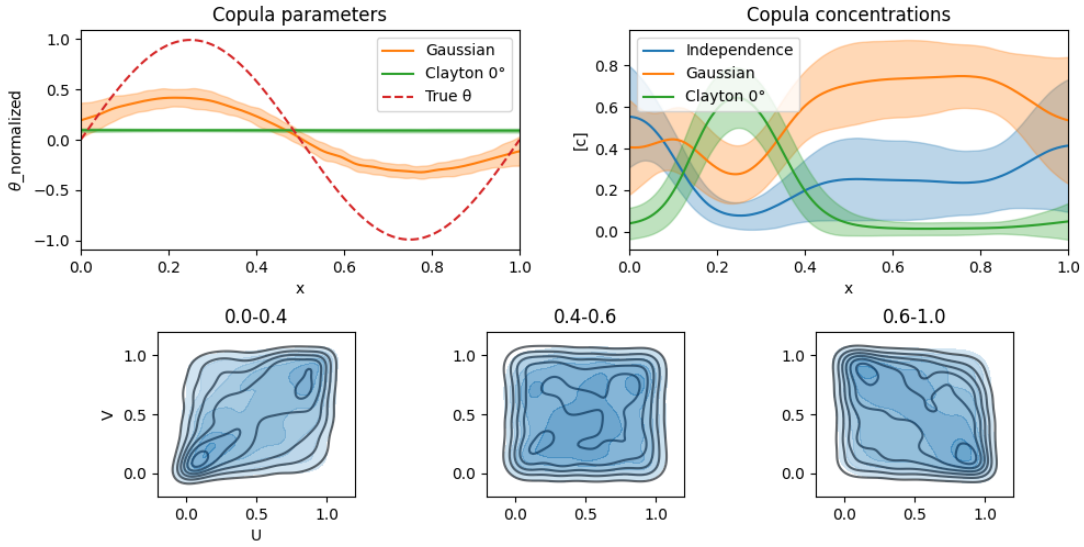

Figure 4.4: Top: GPs of copula parameters and concentrations of the best-fitting copula mixture in the original Copula-GP model. Bottom: The empirical copula density (black) and the estimated copula density (blue) during different intervals of x-values.

COPULA MODEL	WAIC ↓	R^2 ↑
ALI-MIKHAIL-HAQ	-0.0448	$[0.97, 0.98, 0.99]$
INDEPENDENCEGAUSSIANCLAYTON0°	-0.0322	$[0.99, 0.98, 0.99]$

Table 4.1: WAIC and R^2 scores for the AMH copula and the optimal copula mixture model chosen by Copula-GP. The R^2 scores are reported for three intervals: $x \in [0.0, 0.4]$, $x \in [0.4 - 0.6]$, and $x \in [0.6, 1.0]$, in respective order.

mixture must incorporate an additional family, in this case, Clayton, to model the lower tail dependence. This is evident in the GP concentration of the Clayton copula, which remains close to zero except when $x \in [0.2, 0.3]$. Furthermore, as the GP parameter of the Clayton copula remains constant, we conclude that its sole purpose is to capture the tail dependence of the AMH copula.

Lastly, we observe that the Independence copula also makes a notable contribution to the copula mixture, peaking as expected around where θ is close to 0. Since the dependence of the AMH copula is generally weak, the concentration of the Independence copula remains substantial throughout. In particular, the Independence copula accounts for roughly 20% of the mixture between $x = 0.6$ and $x = 0.8$, despite being where the dependence is most negative. It is worth noting that the uncertainties for the Gaussian and Independence copula concentrations are large. This is especially true around $\theta = 0$, as both copulas can individually model independent samples, resulting in some degree of overcompleteness.

We report the WAIC and the R^2 values for the AMH copula and the selected mixture copula in Table 4.1. Our results indicate that despite using data generated from the AMH copula itself, the AMH copula model fails to provide as close of a fit as the mixture model in terms of R^2. While the R^2 scores are generally comparable, the mixture copula outperforms the AMH copula during $x \in [0.0, 0.4]$, where the coefficient

of determination is 99% and 97%, respectively. This result is somewhat surprising but can be attributed to the extreme flexibility of the mixture copula, which is parameterised by a total of 5 GPs. Nevertheless, it is essential to note that the AMH copula does achieve lower WAIC than the mixture copula, which we emphasise as that suggests that the AMH copula is a more parsimonious model which will be preferred in the model selection algorithm.

4.2.2 Evaluating the Joe Copula

In this section, we evaluate the addition of the Joe copula to Copula-GP using synthetic data generated from the R copula package. As the Joe copula has a larger parameter domain of $[1,\infty)$, we modify our ground truth parameter function to cover it accordingly. We draw 9900 samples using the `joeCopula` class by setting $\theta = 3\cos(2\pi x + \pi/2) + 4$ for $x \in [0,1]$. Then, we perform inference using our implementation of the Joe copula on this data.

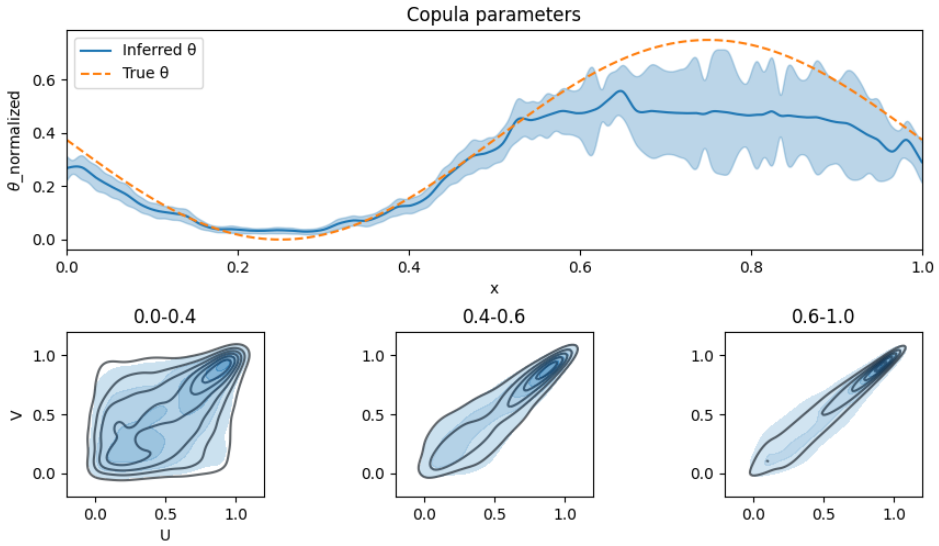

Figure 4.5: Top: Inferred GP parameter of the Joe copula in Copula-GP alongside the true parameter function used to generate the data. Bottom: The empirical copula density (black) and the estimated copula density (blue) during different intervals of x-values.

Figure 4.5 shows the inferred GP parameter alongside the true copula parameter of the Joe copula. We find that the mean function of the GP aligns closely with the ground truth function until $x = 0.6$. Beyond this point, the GP begins to underestimate the value of θ, resulting in a minor mismatch in the empirical copula density and the estimated density. Given the substantial uncertainty in the GP parameter, it is plausible that this discrepancy is not because of a poorly implemented model but rather due to the increased difficulty of inferring accurate estimates as the empirical copula tends to perfect dependence. To validate this hypothesis, we generate similarly distributed data for Clayton and Gumbel copulas, observing that these families also encounter difficulties in differentiating high θ values given the same number of samples. This issue likely emerges because the dependence patterns at large θ-values exhibit minimal

COPULA MODEL	WAIC ↓	R^2 ↑
JOE	−0.6092	[0.94, 0.99, 0.97]
INDEPENDENCECLAYTON180°	−0.6079	[0.91, 0.99, 0.99]
CLAYTON180°	−0.6070	[0.93, 0.99, 0.99]

Table 4.2: WAIC and R^2 scores for the Joe copula and the optimal copula mixture model chosen by Copula-GP. The R^2 scores are reported for three intervals: $x \in [0.0, 0.4]$, $x \in [0.4 - 0.6]$, and $x \in [0.6, 1.0]$, in respective order.

changes when the copula parameter is unbounded and the sample size is limited. To mitigate this, we suggest increasing the number of samples.

Next, we explore the copula mixtures in the original Copula-GP framework that can replicate the dependence structure of the Joe copula. Based on the copula density plots in Figure 4.5, we observe that the Joe copula displays strong upper tail dependence, which is similar to that of the Gumbel and rotated Clayton copulas. We run the heuristics algorithm to select the optimal copula mixture; see Figure 4.6

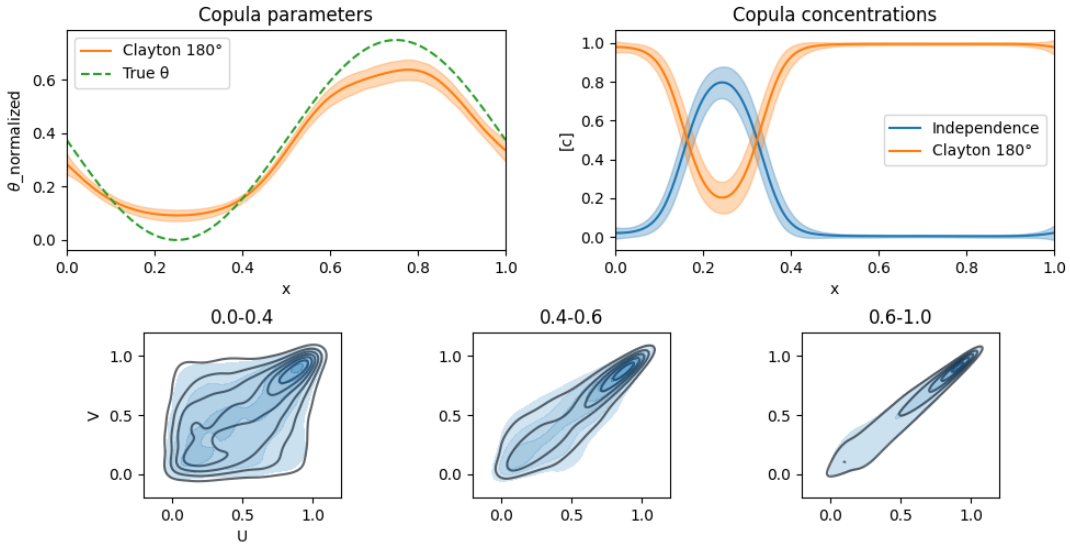

Figure 4.6: Top: GPs of copula parameters and concentrations of the best-fitting copula mixture for reconstructing the Joe copula. Bottom: The empirical copula density (black) and the estimated copula density (blue) during different intervals of x-values.

In Figure 4.6, we observe that the optimal copula mixture consists of the Clayton180°copula and the Independence copula. Interestingly, the Independence copula only contributes to the mixture whenever the ground truth copula reduces to independence. This suggests that the Clayton180°copula may be capable of reconstructing the dependence pattern of the Joe copula by itself. To further validate this, we conduct an additional experiment with just the Clayton180°copula. The WAIC and R^2 scores of our experiments are reported in Table 4.2.

Our results indicate that the Joe copula only achieves minimal improvements in both WAIC and R^2 than the Clayton180°copula and the mixture copula. The poor performance of the Joe copula can be attributed to our previous findings illustrated in Figure

4.5. Since it underestimated the latent parameter function, the estimated conditional copula differed from the true conditional copula, resulting in a lower R^2 score. Moreover, the latent GP of the Joe copula model for $x \geq 0.6$ exhibited large variance, making the final WAIC term being only marginally lower than that of the other models.

4.2.3 Evaluating the Student's t Copula

In this section, we present results from the extended Copula-GP framework using the two-parameter t-copula. The objective of these experiments is to verify the model's ability to simultaneously infer multiple GP parameters using a single copula model. We also seek to examine the extent to which the t-copula's dependence structure can be reconstructed using copula mixtures from the current iteration of Copula-GP and to identify potential limitations of our model. As such, we conduct three experiments, one that showcases the strengths of the t-copula, one that reveals its shortcomings, and one that applies it to high-dimensional data.

In the first experiment, we generate synthetic data using the `ellipCopula()` class from the copula library in R. To parameterise the copula, we use two sinusoidal waves: $\nu = 3\cos(2\pi x + 0.5\pi) + 5$ for the degrees of freedom parameter and $\rho = 0.7 + 0.2\sin(4\pi x)$ for the correlation coefficient where $x \in [0, 1]$. Note that we purposely ensure that the ρ parameter never equals zero, as doing so would interfere with the inference of the ν parameter.

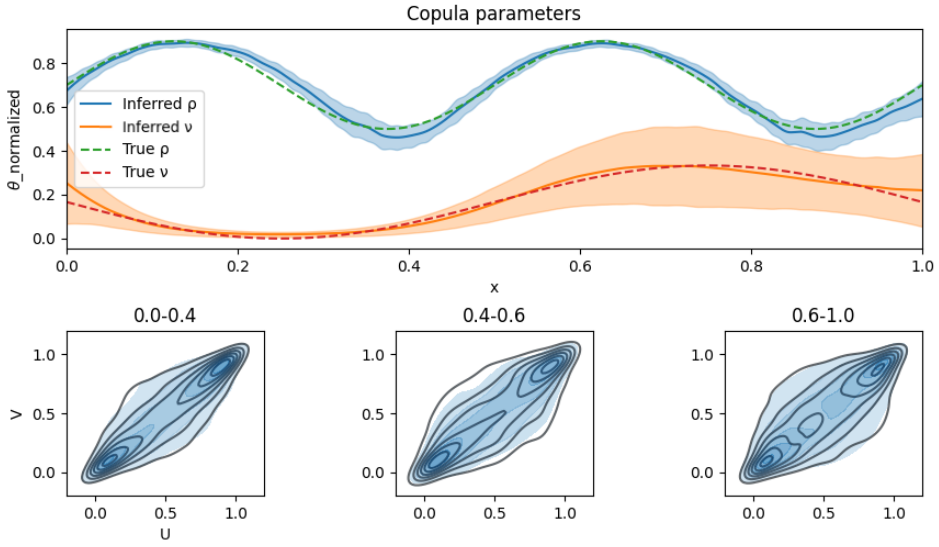

Figure 4.7: Top: Inferred GP parameters of the t-copula in Copula-GP alongside the true parameter functions used to generate the data. Bottom: The empirical copula density (black) and the estimated copula density (blue) during different intervals of x-values.

Figure 4.7 depicts the inferred GP parameters alongside the actual copula parameters of the t-copula. We find that our t-copula model accurately infers the ground truth parameters for both the correlation coefficient and the degrees of freedom. Generally, the uncertainty associated with the correlation coefficient is lower than that of the degrees of freedom since the latter relies on the heavy tails of the copula density, which

COPULA MODEL	WAIC ↓	R^2 ↑
STUDENT'S T	-0.4224	$[0.99, 0.98, 0.99]$
GAUSSIANGUMBEL180°	-0.4156	$[0.98, 0.98, 0.98]$

Table 4.3: WAIC and R^2 scores for the t-copula and the optimal copula mixture model chosen by Copula-GP. The R^2 scores are reported for three intervals: $x \in [0.0, 0.4]$, $x \in [0.4 - 0.6]$, and $x \in [0.6, 1.0]$, in respective order.

is based on a small fraction of samples. The uncertainty for the degrees of freedom is at its lowest at $\nu = 2$ when the t-copula's heavy tails are most prominent. Conversely, the uncertainty for ν rises as its value increases, possibly due to the increased difficulty in inferring confident estimates as the tail dependence weakens (we explore this further in the next experiment).

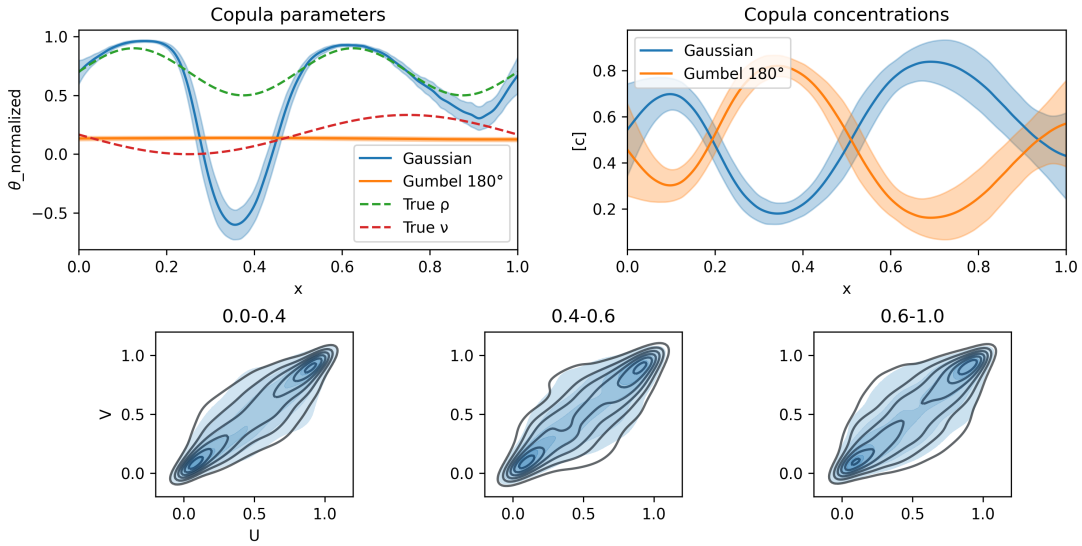

Figure 4.8: Top: Inferred GP parameters of the t-copula in Copula-GP alongside the true parameter functions used to generate the data. Bottom: The empirical copula density (black) and the estimated copula density (blue) during different intervals of x-values.

Using the original Copula-GP package, we find that the optimal copula mixture for modelling the t-copula's dependence pattern consists of the Gaussian and Gumbel180°copulas, see Figure 4.8. This is the copula mixture we used previously in Section 4.1 to tune the convergence rate of the t-copula. The GaussianGumbel°180 mixture is an intuitive approximation to the t-copula as the Gaussian distribution is equivalent to a t-distribution with infinite degrees of freedom. We observe this from the GP parameter of the Gaussian copula, which follows the trajectory of the correlation coefficient to model the copula's interior dependence. As the Gaussian copula does not possess tail dependence, we also find that the mixture utilises the Gumbel copula to replicate the heavy tails of the t-copula. This can be observed in the inverse relationship between the ν and the concentration of the Gumbel copula: when ν is small, the tail dependence is stronger, so the Gumbel copula accounts for more of the mixture to capture the heavy tails. The R^2 and WAIC scores for both copula models are reported in Table 4.3.

4.2.3.1 Limitations of the t-copula

Previously in Section 4.2.2, we pinpointed a limitation of the Joe copula as being unreliable whenever the copula parameter exceeded a certain limit. In this experiment, our objective is to determine if a similar threshold exists for the degrees of freedom parameter of the t-copula. The parameter domain of ν has no upper bound; thus, it can take arbitrarily high values. Nevertheless, in many statistical applications, practitioners often opt for the z-distribution (standard normal) over the t-distribution when the degrees of freedom exceeds 30, as differences beyond this point become negligible.

Considering this, it is plausible that Copula-GP may encounter difficulties when attempting to accurately infer extremely high values of ν. In order to assess the model's performance under such conditions, we designed an experiment where the t-copula's degrees of freedom parameter would progressively grow larger while keeping the sample size the same. To this end, we generate data using the copula library in R [14]. However, this time we keep the correlation coefficient constant and set $\nu = 15\cos(\pi x + 0.5\pi) + 14\pi x + 7$ for $x \in [0,1]$. This yields a parameter range of approximately $\nu = 7$ to $\nu = 50$, see Figure 4.9.

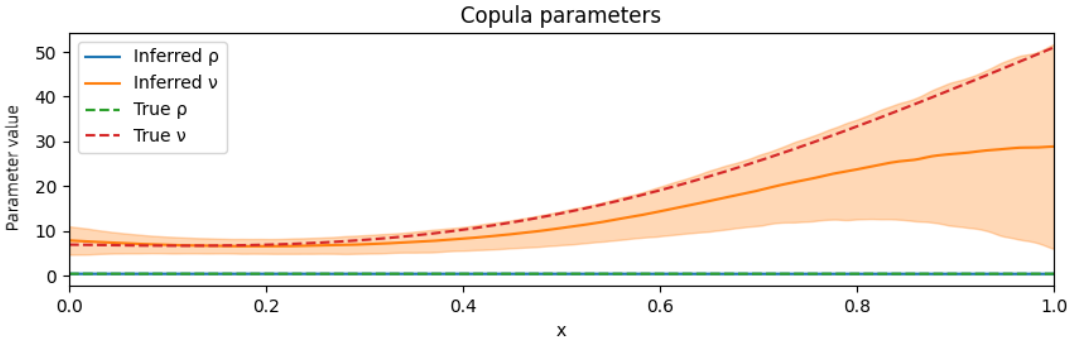

Figure 4.9: The inferred GP parameters alongside the true parameter functions of the t-copula. Note that the parameters in this plot are not normalised for improved visualisation.

In Figure 4.9, we observe that the inferred parameter mostly corresponds to the true parameter value for ν-values less than 10. However, beyond this point, the mean function starts to severely underestimate the degrees of freedom, and this discrepancy only grows larger as the parameter increases. Although the true value of ν always remains within two standard deviations of the MAP estimate (mean function), this demonstrates a significant limitation of the t-copula. Since the ν parameter is only encoded in the tails of the copula, inference for this parameter relies solely on the few samples near the upper-right and lower-left corners of the unit square. As a result, these estimates will be inherently unreliable and will become less confident the more the t-copula tends to the Gaussian copula.

4.2.3.2 Mutual Information Estimation Using the t-copula

As the previous experiments have largely focused on bivariate data, this experiment aims to assess the extended Copula-GP's capability to construct C-vines with copulas

of multiple parameters. To this end, we employ the t-copula and replicate an experiment initially conducted for the original Copula-GP paper [22]. In this experiment, Kudryashova et al. compared Copula-GP against other state-of-the-art mutual information estimators, namely, Kraskov-Stögbauer-Grassberger [21], Bias-Improved-KSG [10], and the Mutual Information Neural Estimator [5]. They generated three datasets, one of which was from the multivariate t-distribution, where the correlation coefficient was fixed at 0.7 and degrees of freedom exponentially increased: $df = e^{5x} + 1, x \in [0, 1]$. Results showed that Copula-GP outperformed all of the aforementioned methods, except for MINE when the data's dependence structure did not correspond to any of the parametric families used in the framework. Therefore, the objective of this experiment is to determine whether the t-copula (which perfectly fits the data) can offer any improvements for mutual information estimation on a high-dimensional dataset.

As described in Section 2.3, Copula-GP provides two methods for mutual information estimation: 'estimated' and 'integrated'. In this experiment, we utilise both methods, modelling both $p(\mathbf{y})$ and $p(\mathbf{y}|x)$ with the t-copula. We then train C-vines for up to eight variables and compare our results from those presented in [22], see Figure 4.10.

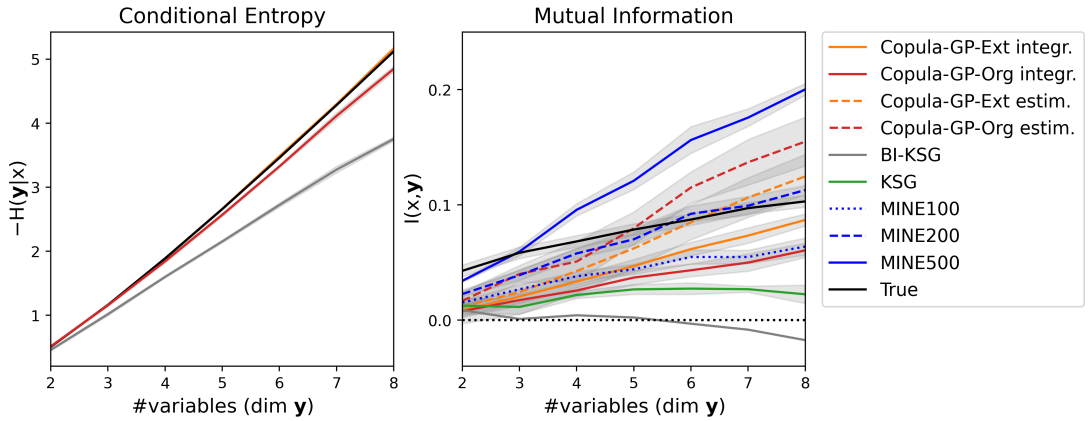

Figure 4.10: **Left**: the conditional entropy $H(\mathbf{y}|x)$ at various $\mathbf{y}$ dimensionalities. MINE estimates are not included in this comparison, as they do not produce estimates of $H(\mathbf{y}|x)$. **Right**: Mutual information $I(x, \mathbf{y})$; the solid and dashed lines for both Copula-GP models represent estimates produced using the 'integrated' and 'estimated' methods, respectively. Note that MINE[X] represents a MINE model with X number of hidden units.

Figure 4.10 shows that the extended Copula-GP provides a near-perfect estimate of conditional entropy, while the original Copula-GP and BI-KSG underestimate the value. Nevertheless, MINE100 outperformed both Copula-GP models in terms of mutual information estimation, despite using copula models with the perfect parametric fit. We also found that similarly to Kudryashova et al. [22], the 'integrated' method underestimates mutual information, while the 'estimated' method overestimates it. Unlike [22], however, our experiments did not show a consistent underestimation of the 'integrated' method. Instead, Figure 4.10 suggests that, at higher dimensions, the estimates provided by the 'integrated' method in the extended Copula-GP will eventually surpass the true mutual information.

Chapter 5

Discussion

In this chapter, we examine the findings obtained from the experiments conducted in Chapter 4 to answer the research questions introduced in Chapter 1. We also discuss any limitations of our work and compare with previous work.

5.1 Research Questions

R1: To what extent does the Ali-Mikhail-Haq copula contribute to the expressive power of the Copula-GP model?

In section 4.2.1, we assessed our implementation of the AMH copula and investigated the dependence pattern exhibited by this family. We found that our AMH model was successful in inferring the latent function used to parameterise the copula from which synthetic data was generated. Crucially, we observed that while the copula mixture from the original framework could reconstruct the dependence pattern of the AMH copula, it required a combination of three copula families to achieve this.

This copula mixture, comprising the Clayton, Gaussian, and Independence copulas, combined to capture the full range of dependence structures present in the AMH copula. Each family served a unique purpose in the mixture: the Gaussian copula addressed the ”central dependence,” the Clayton copula covered the lower tail dependence, and the Independence copula handled weak dependence or independence. Perhaps surprisingly, the combination of copula families even allowed the mixture model to provide a closer fit in terms R^2 than the AMH copula itself. However, as the difference between the two R^2 scores was minimal, further analysis is needed to determine whether this improvement is statistically significant.

In terms of model selection, we discovered that the AMH copula model yielded a significantly lower WAIC than the copula mixture. This finding suggests that, when presented with similar data, the model selection algorithm in Copula-GP will favour the AMH copula over the copula mixture model. This preference is important not only for model parsimony but also for model scalability, as the AMH copula is parameterised by only one GP, whereas the mixture model requires five (two for copula parameters and three for copula concentrations). Since Copula-GP makes use of scalable kernel

interpolation KISS-GP [40], the additional GP required by the copula mixture is not detrimental to training time. However, the increased model complexity of the mixture model may make it more challenging to interpret the underlying dependence structure. Hence, we argue that the unique dependence structure of the AMH copula is a valuable addition to the Copula-GP framework.

R2: To what extent does the Joe copula contribute to the expressive power of the Copula-GP model?

Next, we examined the Joe copula within the context of the Copula-GP. Our results indicated that while the Joe copula model was mostly capable of inferring the general trend of the latent parameter function, these estimates became largely unreliable at high values of the copula parameter. This, in turn, led to minor discrepancies between the empirical and the estimated conditional copulas, affecting both the R^2 scores and WAIC negatively. While we found that other heavy-tailed copula families with unbounded parameter domains (Clayton and Gumbel) also suffered from similar behaviour for highly correlated samples, these families, especially Clayton, proved to be more reliable in discerning large parameter values. This allowed the rotated variant of the Clayton model to achieve comparable results with the Joe copula, despite not being the perfect fit.

The comparable performance between the Joe copula and the rotated Clayton copula suggests that the Joe copula might not provide a substantial advantage over existing copula families within the Copula-GP framework. The ability of the rotated Clayton copula to model the dependence pattern of the Joe copula makes the inclusion of the Joe copula less essential. Nevertheless, as significant efforts have already been made to implement the Joe copula, we propose that it should be included in Copula-GP as an individual model, separate from the model selection algorithm as not to unnecessarily increase its search space. Moreover, it is important to acknowledge that this conclusion is drawn from the specific scenario we examined. In situations where the Joe copula displays a more distinct or complex dependence structure, incorporating it into the framework could yield additional benefits.

R3: To what extent does the Student's *t* copula contribute to the expressive power of the Copula-GP model?

Lastly, we also examined the Student's t copula with two parameters, ρ and ν. Our objective was to verify Copula-GP's ability to simultaneously infer multiple GP parameters using only a single copula. To achieve this, we conducted three experiments, one designed to highlight the strengths of the t-copula, one to reveal its limitations, and one to test it on a high-dimensional dataset. We also assessed the extent to which the t-copula's dependence structure could be reconstructed using copula mixtures from the current iteration of Copula-GP.

In the first experiment, our results showed that the t-copula model within the extended Copula-GP framework was successful in accurately inferring the ground truth parameters for both ρ and ν. This outcome underscores the potential of our proposed architecture to accommodate copula models with multiple parameters and complex dependence structures. Regarding the WAIC and R^2 scores, our t-copula model outperformed the

optimal copula mixture, which comprised Gaussian and Gumbel180°copulas.

In the second experiment, we identified potential limitations of the t-copula, focusing on its challenges in accurately inferring high values of the degrees of freedom parameter (ν). We discovered that our copula model generated highly uncertain estimates, particularly as the degrees of freedom increased. However, we contend that this is not due to a poorly implemented model but rather the increasing difficulty of performing inference when the dependencies are only encoded in the tails. We support this argument with the observation that our copula model did not produce confident yet incorrect estimates but rather uncertain estimates, which, despite their inaccuracy, captured the true underlying function within two standard deviations of the mean. This also further emphasises the advantage of employing a Bayesian approach like Gaussian Processes within the context of Copula-GP.

In the final experiment, our aim was to apply the modified architecture proposed in Section 3.0.5 to construct a C-vine consisting of multiple parameters. For this purpose, we used the t-copula for mutual information estimation on a dataset generated from the multivariate t-distribution. We found that the vine model provided near-perfect estimates for conditional entropy, outperforming both BI-KSG and the previous Copula-GP model. However, for mutual information, we were unable to achieve results as favourable as MINE100 despite having the perfect model fit. As the degrees of freedom for the multivariate t-distribution in this dataset increased exponentially with a maximum value of $e^5 \approx 148$, we hypothesise that our inconsistent results may be due to the issue previously identified in Section 4.9, specifically, the inability to produce confident and accurate estimates when the degrees of freedom are large. This is particularly noteworthy since both experiments were conducted with similar sample sizes.

Despite these limitations, we believe that the Student's t is a valuable addition to the framework as it serves as an excellent starting point for incorporating multi-parameter copulas to Copula-GP.

5.2 Comparison with Alternative Methods

The most notable contribution of this project is the expansion of the Copula-GP framework, which is now able to accommodate copulas with an arbitrary number of parameters. Although Gaussian process copula models using multiple parameters have previously been proposed in work by Hernandez et al. [13], the methodologies behind Copula-GP and that of Hernandez et al. differ in several aspects.

First and foremost, the method proposed in [13] is limited to bivariate data for copulas with multiple parameters. In contrast, our experiments in Section - have demonstrated that multivariate C-vine copulas in Copula-GP can be constructed from the two-parameter t-copula to accurately model dependencies of much higher dimensionality. As a result, the extended Copula-GP framework can use copulas with multiple parameters to model data with multiple dimensions, providing a superior modelling ability compared to that of Hernandez et al.

Secondly, the extended Copula-GP model is designed such that multiple parameter

copulas in Copula-GP may be used as individual models or as part of copula mixtures. This is qualitatively different from [13], where the multi-parameter copulas are limited to being used as individual copula models. This proves to be a significant improvement, as previous work by Kudryashova et al. [**?**] has shown that linear copula mixtures considerably outperform single-element methods in terms of log-likelihood. Moreover, the model selection algorithm in Copula-GP ensures that multi-parameter copulas are used only when required, as they generally yield higher WAIC than single elements unless all parameters contribute significantly to the final model.

Lastly, both [13] and our work uses the Student's t copula as the exemplar multi-parameter copula. However, our t-copula model is fully implemented in PyTorch, allowing for GPU-accelerated computing that effectively reduces the training time by several factors.

5.3 Future work

Validation on real data: Since the primary objective of this project was to add new copula families to Copula-GP, our evaluation mainly focused on determining the extent to which our models contributed to the framework's expressive power. To facilitate evaluation, we, therefore, opted to consider only synthetic data so that we could compare the extended Copula-GP against the original Copula-GP in a controlled setting. Nonetheless, since Copula-GP was originally developed to analyse dependencies within neuronal populations, our examination of each copula without testing on actual neuronal data could be deemed incomplete. As a result, a significant limitation of this work is that we relied exclusively on artificial data. Although the AMH copula has been applied to analyse neuronal data [17], the use of Joe copula and Student's t copula in such settings is less prevalent. It would, therefore, be valuable to explore whether these copula families can provide meaningful insights for analysing neural data.

Incorporating asymmetric copulas. While this project has focused on the Student's t copula as a starting point for extending the Copula-GP framework, there exists an abundance of other multi-parameter copulas. One class of copulas which are of particular interests are asymmetric copulas, which provide a richer set of dependence structures as they exhibit asymmetry along the main diagonal of the unit square. With the modified architecture proposed in Section 3.0.5, incorporating asymmetric copulas within the Copula-GP framework would not require extensive work. This is particularly true since asymmetric copulas can be constructed from Archimedean copulas such as Joe and Clayton [24], which are already included in the framework.

Providing confidence estimates. Our evaluation of the new copula families showed that the WAIC and R^2 scores of the added copula families often differed minimally from those of the copula mixture in the original framework. As the model selection algorithm of Copula-GP is computationally expensive, we opted to run most of our experiments only once, which prevented us from reporting any confidence intervals. We acknowledge this as a major limitation to our work, especially since Copula-GP uses the Adam optimiser for SVI, which are two inherently stochastic methods that may converge to different solutions each time.

Chapter 6

Conclusions

In this thesis, we have investigated the integration of additional copula families to improve the expressive power of the Copula-GP framework (Kudryashova et al.) [22]. We focused on implementing three new parametric copula families: the Ali-Mikhail-Haq copula, the Joe copula, and the Student's *t* copula.

A series of experiments were conducted to evaluate the benefits of incorporating each copula family into the Copula-GP model. We assessed their contributions by examining their ability in providing novel dependence structures across different synthetic datasets. Our experiments indicated that the AMH copula has a unique dependence pattern that can be effectively captured only by combining multiple existing copula families within the framework. This suggests that the inclusion of the AMH copula contributes positively to the overall expressivity of Copula-GP. Conversely, the Joe copula's dependence patterns were accurately reproduced by other copula families, particularly the rotated Clayton copula, rendering the addition of the Joe copula less essential.

The Student's *t* copula emerged as a valuable addition to the Copula-GP model, given its two-parameter structure. Despite some limitations in handling high degrees of freedom, the incorporation of the t-copula underscores the potential for Copula-GP to work with multi-parameter copulas. In our proposed architecture, these copulas can function as standalone models or as part of copula mixtures. Furthermore, we assessed the capability of our models to construct high-dimensional C-vines from multi-parameter copulas and found that the t-copula performed favourably against other state-of-the-art mutual information estimators.

In conclusion, the novelty of this project lies in the extension of the Copula-GP framework to accommodate multi-parameter copulas, representing a significant step towards making Copula-GP a more versatile tool for neural dependence modelling. Our findings demonstrate that the inclusion of additional copula families, particularly those with multiple parameters, has the potential to greatly improve the expressive power of the Copula-GP framework. The inclusion of the Ali-Mikhail-Haq and Student's *t* copulas provides substantial benefits, while the contribution of the Joe copula appears less crucial.

Bibliography

[1] Different correlation structures in copulas. https://datasciencegenie.com/different-correlation-structures-in-copulas/, 2023. Accessed: 2023–04-06.

[2] Kjersti Aas, Claudia Czado, Arnoldo Frigessi, and Henrik Bakken. Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics*, 44(2):182–198, 2009.

[3] Mir M Ali, N.N Mikhail, and M.Safiul Haq. A class of bivariate distributions including the bivariate logistic. *Journal of Multivariate Analysis*, 8(3):405–412, 1978.

[4] T.J. Bedford and R. Cooke. Monte carlo simulation of vine dependent random variables for applications in uncertainty analysis. 2001. ESREL 2003 ; Conference date: 15-06-2003 Through 18-06-2003.

[5] Ishmael Belghazi, Sai Rajeswar, Aristide Baratin, R. Devon Hjelm, and Aaron C. Courville. MINE: mutual information neural estimation. *CoRR*, abs/1801.04062, 2018.

[6] Pietro Berkes, Frank Wood, and Jonathan Pillow. Characterizing neural dependencies with copula models. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

[7] Edwin V Bonilla, Kian Chai, and Christopher Williams. Multi-task gaussian process prediction. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[8] Jason Catlett. Statlog (shuttle) data set. https://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle), 2023. Accessed: 2023–04-06.

[9] Peng Ding. On the conditional distribution of the multivariate *t* distribution, 2016.

[10] Weihao Gao, Sewoong Oh, and Pramod Viswanath. Demystifying fixed k-nearest neighbor information estimators. *IEEE Transactions on Information Theory*, PP:1–1, 02 2018.

[11] Jochen Görtler, Rebecca Kehlbeck, and Oliver Deussen. A visual exploration of gaussian processes. *Distill*, 2019. https://distill.pub/2019/visual-exploration-gaussian-processes.

[12] Martin Haugh. An introduction to copulas, 2016. Available at: http://www.columbia.edu/ mh2078/QRM/Copulas.pdf. Accessed: 18-10-2022.

[13] José Miguel Hernández-Lobato, James R Lloyd, and Daniel Hernández-Lobato. Gaussian process conditional copulas with applications to financial time series. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[14] Marius Hofert, Ivan Kojadinovic, Martin Maechler, and Jun Yan. *copula: Multivariate Dependence with Copulas*, 2023. R package version 1.1-2.

[15] Murray Iain, Onken Arno, and Vergari Antonio. Gaussian processes. Available at: https://mlpr.inf.ed.ac.uk/2022/notes/w5b_gaussian_processes.html. Accessed: 20-10-2022.

[16] Institute and Faculty of Actuaries. Giro40. https://www.actuaries.org.uk/system/files/documents/pdf/b3-presentation.pdf. Accessed: 2023–04-06.

[17] Rick Jenison and Richard Reale. The shape of neural dependence. *Neural computation*, 16:665–72, 05 2004.

[18] Harry Joe. Asymptotic efficiency of the two-stage estimation method for copula-based models. *Journal of Multivariate Analysis*, 94(2):401–419, 2005.

[19] Harry Joe. *Dependence Modeling with Copulas*. Chapman and Hall/CRC New York, NY, 2014.

[20] Qingkai Kong, Timmy Siauw, and Alexandre M. Bayen. Chapter 19 - root finding. In Qingkai Kong, Timmy Siauw, and Alexandre M. Bayen, editors, *Python Programming and Numerical Methods*, pages 325–335. Academic Press, 2021.

[21] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Phys. Rev. E*, 69:066138, Jun 2004.

[22] Nina Kudryashova, Theoklitos Amvrosiadis, Nathalie Dupuy, Nathalie Rochefort, and Arno Onken. Parametric copula-gp model for analyzing multidimensional neuronal and behavioral relationships. *PLOS Computational Biology*, 18(1):1–30, 01 2022.

[23] Pranesh Kumar. Probability distributions and estimation of ali-mikhail-haq copula. *Applied Mathematical Sciences*, 4:657–666, 01 2010.

[24] Eckhard Liebscher. Construction of asymmetric multivariate copulas. *Journal of Multivariate Analysis*, 99(10):2234–2250, 2008.

[25] David Lopez-Paz, Jose Miguel Hernández-Lobato, and Zoubin Ghahramani. Gaussian process vine copulas for multivariate dependence. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page II–10–II–18. JMLR.org, 2013.

[26] Lazaros Mitskopoulos, Theoklitos Amvrosiadis, and Arno Onken. Mixed vine copula flows for flexible modeling of neural dependencies. *Frontiers in Neuroscience*, 16, 2022.

[27] S.L.B. Moshier. *Methods and Programs for Mathematical Functions*. Ellis Horwood series in mathematics and its applications. E. Horwood, 1989.

[28] Roger B. Nelsen. *An Introduction to Copulas*. Springer New York, NY, 2006.

[29] Arno Onken, Steffen Grünewälder, Matthias Munk, and Klaus Obermayer. Modeling short-term noise dependence of spike counts in macaque prefrontal cortex. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

[30] Arno Onken, Steffen Grünewälder, Matthias H. J. Munk, and Klaus Obermayer. Analyzing short-term noise dependencies of spike-counts in macaque prefrontal cortex using copulas and the flashlight transformation. *PLOS Computational Biology*, 5(11):1–13, 11 2009.

[31] Arno Onken and Stefano Panzeri. Mixed vine copulas as joint models of spike counts and local field potentials. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[32] Travis A. O'Brien, Karthik Kashinath, Nicholas R. Cavanaugh, William D. Collins, and John P. O'Brien. A fast and objective multidimensional kernel density estimation method: fastkde. *Computational Statistics Data Analysis*, 101:148–160, 2016.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[34] Carl Edward Rasmussen. Gaussian processes in machine learning. Available at: http://mlg.eng.cam.ac.uk/pub/pdf/Ras04.pdf. Accessed: 20-10-2022.

[35] Laura Sacerdote, Massimiliano Tamborrino, and Cristina Zucca. Detecting dependencies between spike trains of pairs of neurons through copulas. *Brain Research*, 1434:243–256, 2012. Selected papers presented at the International Workshop on Neural Coding, Limassol, Cyprus, 29 October - 3 November 2010.

[36] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan

Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[37] Eric Weisstein. Erf. https://mathworld.wolfram.com/Erf.html, 2002. Accessed: 2023–04-06.

[38] Eric Weisstein. Gamma function. https://mathworld.wolfram.com/GammaFunction.html, 2002. Accessed: 2023–04-06.

[39] Eric Weisstein. Householder's method. https://mathworld.wolfram.com/Erf.html, 2002. Accessed: 2023–04-06.

[40] Andrew Gordon Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp), 2015.

Appendix A

Runtime Experiments

A.1 Newton-Raphson's Method or Halley's Method?

In Section 3.0.3.2, we considered two options for numerically approximating the inverse Joe conditional copula, one using the Newton-Raphson method and another using the Halley's method. To test the performance of both methods, we generated 10^6 samples and transformed them with the Joe conditional copula. Then, we checked whether the inverse function could correctly invert all samples given a distance tolerance of 10^{-5}. The results are presented in Table A.1.

Table A.1: Runtime and convergence statistics for Newton and Halley's methods in the PPCF experiment.

	Halley's Method		Newton's Method	
Iterations	Runtime (s)	Outcome	Runtime (s)	Outcome
9	0.612	Fail	0.478	Fail
10	0.621	Success	0.548	Fail
11	0.755	Success	0.618	Success
12	0.755	Success	0.626	Success
13	0.800	Success	0.600	Success
14	0.939	Success	0.642	Success

As shown in Table A.1, the fastest method was Newton's Method. Although Halley's method required fewer iterations, the additional computational costs of computing the second derivative offset its faster convergence. Therefore, we decided to implement the inverse Joe conditional copula using the Newton-Raphson's method.

Appendix B

Code Implementation

B.1 Implementing the Student's t distribution CDF and PPF

B.1.1 Regularised Incomplete Beta Integral (Beta CDF)

```
import torch
import pdb
'''
Implementation of the incomplete beta function and its inverse. For computing the CDF and ICDF of the student's t-distribution.
'''
def betaln(a, b):
    return torch.lgamma(a) + torch.lgamma(b) - torch.lgamma(a + b)

def betainc(a, b, x):
    device = x.device
    a = a.clone()
    b = b.clone()
    x = x.clone()

    # edge cases
    nan_mask = (a < 0.0) | (b < 0.0)
    zero_mask = x < 0.0
    one_mask = x > 1.0

    inversion_mask = x > (a + 1.0) / (a + b + 2.0)
    a[inversion_mask], b[inversion_mask] = b[inversion_mask], a[inversion_mask]
    x[inversion_mask] = 1.0 - x[inversion_mask]


    front = torch.exp(torch.log(x) * a + torch.log(1.0 - x) * b - betaln(a, b)) / a

    # constants
    EPSILON = 1e-30
    THRESH = 1e-5
```

```
    MAX_ITER = 10

    f = torch.ones_like(a)
    c = torch.ones_like(a)
    d = torch.zeros_like(a)

    # iterate until convergence
    for i in range(MAX_ITER):
        m = i // 2

        numerator = torch.where(torch.tensor([i == 0], dtype=torch.
            bool).to(device),
                        torch.ones_like(a),
                        torch.where(torch.tensor([i % 2 == 0], dtype
                            =torch.bool).to(a.device),
                                    (m * (b - m) * x) / ((a + 2.0 *
                                        m - 1.0) * (a + 2.0 * m)),
                                    -((a + m) * (a + b + m) * x) /
                                        ((a + 2.0 * m) * (a + 2.0 * m
                                         + 1))))


        d = 1.0 + numerator * d
        d = torch.where(torch.abs(d) < EPSILON, torch.tensor(EPSILON
            ).to(a.device), d)
        d = 1.0 / d

        c = 1.0 + numerator / c
        c = torch.where(torch.abs(c) < EPSILON, torch.tensor(EPSILON
            ).to(a.device), c)

        cd = c * d
        f *= cd

        if torch.all(torch.abs(1.0 - cd) < THRESH):
            break

    result = torch.where(inversion_mask, 1 - front * (f - 1.0),
        front * (f - 1.0))

    # put any out of range values to their respective values
    result[nan_mask] = float('nan')
    result[zero_mask] = 0.0
    result[one_mask] = 1.0

    return result
```

Listing B.1: Python code which implements the regularised incomplete beta integral for the computation of the beta CDF.

B.1.2 Student's t CDF and PPF

```
import torch
from . import beta_distribution as beta
import pdb
```

```
import torch.distributions.studentT as studentT
import torch.distributions.normal as normal

def t_cdf(x, df):
    x = x.unsqueeze(0)
    aa = torch.ones_like(x) * 0.5
    df = df.expand_as(x)
    beta_val = beta.betainc(aa, df/2, (x*x)/(df + x*x))
    vals = torch.where(x < 0, 0.5 * (1 - beta_val), 0.5 * (1 +
        beta_val))
    return vals.squeeze()


def t_deriv(x, df):
    val = studentT.StudentT(df).log_prob(x)
    return torch.exp(val)

def t_ppf(p, df, max_iter=4, tol=1e-5):
    p = p.clone()
    # use the normal ppf as an initial guess
    x = normal.Normal(torch.zeros(1, device=p.device), torch.ones(1,
        device=p.device)).icdf(p)
    for _ in range(max_iter):
        # evaluate the function (CDF) and its derivative (PDF) at
            the current guess
        f = t_cdf(x, df) - p
        f_prime = t_deriv(x,df)

        # set x_new to positive or negative infinity when f_prime is
            zero
        x_new = torch.where(
            (x == float('inf')) | (x == float('-inf')),
            x,
            x - f / f_prime
        )

        # check for convergence
        if torch.all(torch.abs(x_new - x) < tol):
            break

        x = x_new
    return x
```

Listing B.2: Python code which implements the Student's t distribution CDF and PPF using the regularised incomplete beta integral.