CQE Center for Quantitative Economics

Bayesian Estimation of Generalized Partition of Unity Copulas

Andreas Masuhr[†]

73/2018

[†] Department of Economics, University of Münster, Germany

wissen.leben
WWU Münster

Bayesian Estimation of Generalized Partition of Unity Copulas

Andreas Masuhr

Westfälische Wilhelms-Universität Münster

June 1, 2018

Abstract

This paper proposes two (Metropolis-Hastings) algorithms to estimate Generalized Partition of Unity Copulas (GPUC), a new class of nonparametric copulas that includes the versatile Bernstein Copula as a special case. Additionally a prior distribution for the parameter Matrix of GPUCs is established via Importance Sampling and an algorithm to sample such matrices is introduced. Finally, simulation studies show the effectiveness of the presented algorithms.

1 Introduction

A bivariate copula is a function $C : [0,1]^2 \to [0,1]$ with the following properties[1]:

1. $C(u,1) = C(0,v) = 0$
2. $C(u,1)u$; $C(1,v) = v$
3. $C(u,v)$ is 2-increasing, i.e. for $u_1,\ u_2$ and $v_1,\ v_2$ with $u_1 \leq u_2$ and $v_1 \leq v_2$ it holds:

$$C(u_2,v_2) - C(u_2,v_1) - C(u_1,v_2) + C(u_1,v_1) \geq 0.$$

Then, using Sklar's theorem, any bivariate distribution $F_{XY}(x,y)$ can be split up into its marginal distributions $F_X(x)$ and $F_Y(y)$ and its copula $C(u,v)$, with $u = F_X(x)$ and $v = F_Y(y)$ the following way:

$$F_{XY}(x,y) = C\big(F_X(x), F_Y(y)\big). \tag{1}$$

If X and Y are continuous random variables, then the copula $C(u,v)$ is unique. Using this representation allows to treat the marginal distributions and their dependence structure, separately and hence, adds more flexibility to the modeling process. Various families of copulas have emerged in the past years ranging from the *classic* families that are generated by inversion of multivariate distributions like the Gaussian or t-copula to the class of Archimedean copulas that use certain generator functions to create copulas. To add even more flexibility in higher dimensions all these copula families might be mixed using vines (Bedford and Cooke (2002)) or, within the class of Archimedean copulas bivariate copulas can be organized hierarchically (Savu and Trede (2010)). Yet another way to construct copulas is to use a nonparametric approach by trying to directly mimic the characteristics of the data. The simplest way of doing so is to just use the empirical copula $C^*(u,v)$:

$$C^*(u,v) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{(U_i<u, V_i<v)}.$$

Another nonparametric copula is the Bernstein Copula Baker (2008)). This copula uses Bernstein polynomials to smoothly approximate the copula density. One

[1] For more on copulas, see Nelsen (2006)

method to estimate Bernstein Copulas is an EM-Algorithm, proposed by Dou et al. (2016). Recently, another family of nonparametric copulas has been developed, the so called Generalized Partition of Unity Copulas (GPUC; Pfeifer et al. (2016)). In contrast to the Bernstein Copula the unit hypercube is no longer approximated by a finite mixture of functions but by an infinite mixture. As a special case GPUCs also nest the Bernstein Copula.
This paper proposes a Bayesian MCMC based method to estimate GPUCs and proceeds as follows: the second section introduces the Generalized Partition of Unity Copulas. Chapter three describes an algorithm that allows to sample parameter matrices of Generalized Partition of Unity Copulas. This algorithm is used in section four to construct a prior distribution for the aforementioned parameter matrices. Section five in turn presents two MCMC algorithms to sample from the posterior distribution of these parameters. The two algorithms are compared based on simulated data in section six and section seven concludes.

2 Generalized Partition of Unity Copulas

The concept of Generalized Partition of Unity Copulas (GPUC) was first introduced by Pfeifer et al. (2016) and applied in the context of risk management Pfeifer et al. (2017). A GPUC is a copula defined by the density function:

$$c(u,v) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} M_{ij}\frac{\phi_i(u)}{\alpha_i}\frac{\varphi_j(v)}{\gamma_j}, \tag{2}$$

with $\alpha_i = \sum_{j=0}^{\infty} M_{ij} = \int_0^1 \phi_i(u)du$ and $\gamma_j = \sum_{i=0}^{\infty} M_{ij} = \int_0^1 \varphi_j(v)dv$. The *generating functions* $\phi_i(u)$ and $\varphi_j(v)$ can be considered as probability mass functions of discrete random variables over the non-negative integers $\mathbb{Z}^+$ with parameters u and v, respectively and hence, $\sum_{i=0}^{\infty}\phi_i(u) = \sum_{j=0}^{\infty}\varphi(v) = 1$. In the subsequent paper I only consider bivariate GPUCs with $\phi_i(u) = \varphi_j(u)$ for $i = j$, i.e. copulas with identical generating functions. Equation (2) denotes the fully parametrized form GPUC with an infinitely large parameter matrix $\boldsymbol{M}$. For estimation, I consider a reduced form GPUC, i.e.

$$c(u,v) = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} M_{ij}\frac{\phi_i(u)}{\alpha_i}\frac{\varphi_j(v)}{\gamma_j} + \sum_{i=m}^{\infty}\frac{\phi_i(u)\varphi_i(v)}{\alpha_i}. \tag{3}$$

This form requires a $m \times m$ matrix to parameterize the lower left part of $[0,1]^2$. Accordingly, the upper right part can be parametrized by additional parameters of the generating functions $\phi_i(u)$ and $\varphi_j(v)$.

Proposition 2.0.1. *All fully parameterized GPUCs nest the independence copula if* $M_{ij} = \alpha_i \gamma_j \; \forall i, \; j$

This property of GPUCs can easily be verified as

$$c(u,v) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} M_{ij} \frac{\phi_i(u)}{\alpha_i} \frac{\varphi_j(v)}{\gamma_j} = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \phi_i(u) \varphi_j(v) = 1$$

Consequently, all reduced form GPUCs don't include the independence copula anymore and hence care must be taken when specifying the size of the parameter Matrix $\boldsymbol{M}$ before estimation. As an example of a GPUC, consider the probability mass function of a negative binomial random variable as generating function:

$$\phi_i(u) = \binom{\beta + i - 1}{i} (1-u)^{\beta} u^i.$$

Therefore,

$$\alpha_i = \int_0^1 \phi_i(u) du = \int_0^1 \binom{\beta + i - 1}{i} (1-u)^{\beta} u^i du = \frac{\beta}{(\beta + i)(\beta + i + 1)}$$

and finally the density is given by,

$$c(u,v) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \frac{M_{ij} \beta^2 \binom{\beta+i+1}{i} \binom{\beta+j+1}{j} (1-u)^{\beta} (1-v)^{\beta} u^i v^j}{(\beta+1)^2 (\beta+i+1)(\beta+j+1)} \tag{4}$$

for the full form and

$$\begin{aligned} c(u,v) = &\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \frac{M_{ij} \beta^2 \binom{\beta+i+1}{i} \binom{\beta+j+1}{j} (1-u)^{\beta} (1-v)^{\beta} u^i v^j}{(\beta+1)^2 (\beta+i+1)(\beta+j+1)} \\ &+ \sum_{i=m}^{\infty} \frac{(\beta+1)(\beta+i+1)}{\beta} \binom{\beta + i - 1}{i}^2 (1-u)^{\beta} (1-v)^{\beta} u^i v^i \end{aligned} \tag{5}$$

for the reduced form GPUC. Using negative binomial distributions as generating functions results in a copula with upper tail dependence, according to Pfeifer et al. (2016).

By using binomial distributions as generating functions, the Bernstein Copula is

obtained. Note, that both negative Binomial GPUC and Bernstein Copula can be represented by mixtures of Beta distributions, offering an appealing way to simulate them (see appendix). The aim of this paper is to estimate the parameter matrix of a reduced form GPUC, i.e. a matrix $\boldsymbol{M}$ from the space

$$\mathcal{M}_\beta = \left\{ \boldsymbol{M} : M_{ij} \geq 0 \forall i,j;\ \sum_{i=1}^{m} M_{i,j} = \alpha_j;\ \sum_{j=1}^{m} M_{i,j} = \alpha_i \right\}. \tag{6}$$

Accordingly, one can define the set $\mathcal{G}_\beta$ of $(m-1) \times (m-1)$ GPUC matrix candidates by

$$\begin{aligned} \mathcal{G}_\beta = \Bigg\{ \boldsymbol{G} : G_{ij} \geq 0 \forall i,j;\ \alpha_i - \alpha_m \leq \sum_{j=1}^{m-1} G_{ij} \leq \alpha_i;\ \alpha_j - \alpha_m \leq \sum_{i=0}^{m-1} G_{ij} \leq \alpha_j; \\ \sum_{k=1}^{m-1} \alpha_k - \alpha_m \leq \sum_{i=1}^{m-1} \sum_{j=1}^{m-1} G_{ij} \leq \sum_{k=1}^{m-1} \alpha_k \Bigg\}. \end{aligned} \tag{7}$$

$\mathcal{G}_\beta$ includes all $(m-1) \times (m-1)$ matrices that can be expanded into eligible GPUC parameter matrices by adding an m-th row and m-th column to $\boldsymbol{G}$ and setting the values of these row and column equal to

$$G_{i,m} = \alpha_i - \sum_{j=1}^{m-1} G_{i,j} \text{ and } G_{m,j} = \alpha_j - \sum_{i=1}^{m-1} G_{i,j}.$$

3 Sampling GPUC Matrices

In order to use a MCMC-based sampling scheme this section proposes an algorithm to draw from the space of (reduced form) GPUC parameter matrices. To that end, the matrices $\boldsymbol{M}^+$ and $\boldsymbol{M}^-$ that trace upper and lower limits of the parameter matrix $\boldsymbol{M}$ that are initialized as:

$$\boldsymbol{M}_0^+|\beta = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_m \\ \alpha_2 & \alpha_2 & \alpha_3 & \dots & \alpha_m \\ \alpha_3 & \alpha_3 & \alpha_3 & \dots & \alpha_m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_m & \alpha_m & \alpha_m & \dots & \alpha_m \end{bmatrix} \text{ and } \boldsymbol{M}_0^-|\beta = \begin{bmatrix} \max\{\alpha_1 - \sum_{j=2}^{m} \alpha_j, 0\} & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}. \tag{8}$$

Now, to sample a parameter matrix M, the following algorithm is proposed:

Initialize $\boldsymbol{M} = \mathbf{0}$;
Initialize $m \times m$ matrices $\boldsymbol{R} = \mathbf{0}$ and $\boldsymbol{C} = \mathbf{0}$;
Initialize $\boldsymbol{M}_0^+$ and $\boldsymbol{M}_0^-$ as in (8);
for *i in 1:m* **do**
  for *j in 1:m* **do**
    Draw $m_{i,j}$ uniformly distributed on $[M_{ij}^-, M_{ij}^+]$;
    Compute $r_i = \alpha_i - \sum_{k=1}^{j} \boldsymbol{M}_{ik}$ and $c_j = \alpha_j - \sum_{k=1}^{i} \boldsymbol{M}_{kj}$;
    Set $\boldsymbol{R}_{ij} = r_i$ and $\boldsymbol{C}_{ij} = c_j$;
    Update $\boldsymbol{M}^+$ and $\boldsymbol{M}^-$ according to:

- $\boldsymbol{M}^+ = pmin\{\boldsymbol{R}, \boldsymbol{C}\}$, where pmin is the element wise minimum;
- $\boldsymbol{M}_{ij}^- = max\left\{\alpha_i - \sum_{k=1}^{j} \boldsymbol{M}_{ik} - \sum_{k=j+1}^{m} \boldsymbol{M}_{ik}^+,\ \alpha_j - \sum_{k=1}^{i} \boldsymbol{M}_{kj} - \sum_{k=i+1}^{m} \boldsymbol{M}_{kj}^+,\ 0\right\} \forall i, j;$

  end
end

Algorithm 1: Sampling GPUC parameter matrices.

The proof that all matrices generated with this algorithm are GPUC parameter matrices follows by contradiction: Assume a generated matrix $\widetilde{\boldsymbol{M}}$ is not a GPUC matrix. This might happen in three cases:

1. $\widetilde{\boldsymbol{M}}_{ij} < 0$ for some i, j.
2. $\sum_{k=1}^{m} \widetilde{\boldsymbol{M}}_{kj} \neq \alpha_j$ or $\sum_{k=1}^{m} \widetilde{\boldsymbol{M}}_{ik} \neq \alpha_i$ for some i, j
3. $\sum_{i=1}^{m} \sum_{j=1}^{m} \widetilde{\boldsymbol{M}}_{ij} \neq \sum_{j=1}^{m} \alpha_j = \sum_{i=1}^{m} \alpha_i$

Now, it is sufficient to show that the first case cannot happen and that the subsequent cases would require the former one(s) to happen: For all elements of $\widetilde{\boldsymbol{M}}$ the first case cannot occur, since the minimum $\boldsymbol{M}_{ij}^-$ in each draw is limited from below by 0. The maximum of each cell is non negative as well, which again is shown by contradiction: Asssume $\boldsymbol{M}_{ij}^+$ is negative for the first time in the algorithm. Consequently, either $\alpha_i < \sum_{k=1}^{j-1} \boldsymbol{M}_{ik}$ or $\alpha_j < \sum_{k=1}^{i-1} \boldsymbol{M}_{kj}$ which requires either $\boldsymbol{M}_{i-1,j}^+ > \alpha_j - \sum_{k=1}^{i-2} M_{kj}$ or $\boldsymbol{M}_{i,j-1}^+ > \alpha_i - \sum_{k=1}^{j-2} M_{ik}$, which in turn is ruled out by the definition of the matrices $\boldsymbol{R}$ and $\boldsymbol{C}$.

If $\widetilde{\boldsymbol{M}}_{i,j} \geq 0 \forall i, j$, then by definition of $\boldsymbol{R}$ and $\boldsymbol{C}$ it follows that $\widetilde{\boldsymbol{M}}_{i,m} = \alpha_i - \sum_{k=1}^{m-1} \widetilde{\boldsymbol{M}} i, k$ and hence the second case cannot happen, either. Consequently, it is trivial that the third case cannot happen as well, and hence the algorithm always generates eligible GPUC parameter matrices.
Note, however that this is only the case if the matrices $\boldsymbol{M}^+$ and $\boldsymbol{M}^-$ are initialized as in (8).

4 A Prior distribution on M

The aim of any Bayesian estimation method is to obtain the posterior distribution of the parameters of interest (or at least a sample of it), given the data. In the case of a GPUC, the posterior distribtuion is $p\big(\boldsymbol{M}, \beta | u, v\big) \propto p\big(\boldsymbol{M}, \beta\big) p\big(u, v | \boldsymbol{M}, \beta\big)$. Now, the joint prior of $\boldsymbol{M}$ and β, $p(\boldsymbol{M}, \beta)$ can be factorized into $p(\boldsymbol{M}, \beta) = p(\beta) \cdot p(\boldsymbol{M}|\beta)$. If both, the prior for β and the conditional prior for $\boldsymbol{M}$ are assumed to be flat, the challenge is to identify the volume of the space that $\boldsymbol{M}$ can be sampled from, given β in order to determine the value of the conditional prior of $\boldsymbol{M}$.
To illustrate this consider the simplest case possible, i.e. estimating a 2×2 matrix $\boldsymbol{M}$. In this case there is only one free parameter, $\boldsymbol{M}_{1,1}$ with $\boldsymbol{M}_{1,2}, \boldsymbol{M}_{2,1}$ and $\boldsymbol{M}_{2,2}$ being computed by the constraints on the row and column sums of $\boldsymbol{M}$. Consequently $\boldsymbol{M}_{1,1}$ is drawn from the interval $[\alpha_1 - \alpha_2, \alpha_1]$.[2] Now, generally, α_i depends on β, for example in the case of a Negative Binomial GPUC: $\alpha_i = \frac{\beta}{(\beta+i-1)(\beta+i)}$ and thus, an uninformative prior for a 2×2 matrix $\boldsymbol{M}$ given β is a function of β with $p(\boldsymbol{M}|\beta) = \beta + 3 + \frac{2}{\beta}$.
For arbitrary values of m, a conditionally flat prior can be described by introducing

[2] That is, because $\boldsymbol{M}^+_{1,1} = \alpha_1$ and $\boldsymbol{M}^-_{1,1} = \alpha_1 - \alpha_2$

an auxiliary $(m-1)\times(m-1)$ random matrix $\boldsymbol{B}|\beta \sim U(\boldsymbol{0}, \boldsymbol{M}_0^+)$, where $\boldsymbol{M}_0^+|\beta$ is defined as in (8). Now, the prior for a $m \times m$ matrix $\boldsymbol{M}$ is given by

$$\begin{aligned} p(\boldsymbol{M}|\beta) &= \left(\prod_{i=1}^{m-1} \prod_{j=1}^{m-1} \boldsymbol{M}_{0;i,j}^+ \int \mathbf{1}_{\{\boldsymbol{B}\in\mathcal{G}\}} \left(\prod_{i=1}^{m-1} \prod_{j=1}^{m-1} \boldsymbol{M}_{i,j}^+ \right)^{-1} d\boldsymbol{B} \right)^{-1}, \\ &= \left(E_{\boldsymbol{B}} \left[\mathbf{1}_{\{\boldsymbol{B}\in\mathcal{G}\}} | \beta \right] \prod_{i=1}^{m-1} \prod_{j=1}^{m-1} \boldsymbol{M}_{i,j}^+ \right)^{-1} \\ &= \frac{f(\boldsymbol{B}|\beta)}{E_{\boldsymbol{B}} \left[\mathbf{1}_{\{\boldsymbol{B}\in\mathcal{G}\}} | \beta \right]} \end{aligned} \tag{9}$$

where $\mathcal{G}$ is the set of $(m-1)\times(m-1)$ matrices that are eligible candidates for $m \times m$ GPUC parameter matrices as defined in (7) and $\mathbf{1}_{\{\boldsymbol{B}\in\mathcal{G}\}}$ denotes the indicator function that is equal to one if $\boldsymbol{B} \in \mathcal{G}$ and 0, elsewhere. Evaluating this integral for large m gets very cumbersome using classical approximation schemes due to the high dimensionality.

As a solution, an importance sampling approach is proposed to estimate the above expectation. Following Greenberg (2008), an estimate of $E\big[g(x)\big] = \int g(x)f(x)dx$ with $X \sim F(\cdot)$ and density $f(x)$ can be computed considering the integral

$$E\big[g(x)\big] = \int \frac{g(x)f(x)}{h(x)} h(x)dx$$

and approximating it by a sample of sample of N values X_n from the importance distribution $h(x)$ and computing

$$E\big[g(x)\big] \approx \frac{1}{N} \sum_{n=1}^{N} g(x_n) \frac{f(x_n)}{h(x_n)}.$$

Translated into the setting of GPUCs, $p(\boldsymbol{M}|\beta)$ is obtained by drawing a sample of N values $\boldsymbol{B}_n$ from an importance distribution $h(\boldsymbol{B})$ and then the expectation is approximated by a weighted average over the drawn sample:

$$E_{\boldsymbol{B}}[\mathbf{1}_{\{\boldsymbol{B}\in\mathcal{G}\}}|\beta]^{-1} \approx \left(\frac{1}{N} \sum_{n=1}^{N} \mathbf{1}_{\{\boldsymbol{B}_n\in\mathcal{G}\}} \frac{f(\boldsymbol{B}_n|\beta)}{h(\boldsymbol{B}_n|\beta)} \right)^{-1}, \tag{10}$$

where $f(\boldsymbol{B}_n|\beta) = \left(\prod_{i=1}^{m-1} \prod_{j=1}^{m-1} \boldsymbol{M}_{i,j}^+ \right)^{-1}$. Here, the importance distribution $h(\boldsymbol{B}|\beta)$ is chosen according to the sampling scheme of algorithm 1. Formally,

drawing a candidate $\boldsymbol{B}$ from $h(\boldsymbol{B}|\beta)$ is split into drawing from a set of conditional distributions when sampling a candidate GPUC $\boldsymbol{B}$ matrix, i.e.:

$$\begin{aligned} h(\boldsymbol{B}|\beta) =& p(B_{1,1}) \cdot p(B_{1,2}|M_{1,1}) \cdot \ldots \cdot p(B_{1,m-1}|B_{1,1}, \ldots, B_{1,m-2}) \\ & \cdot p(B_{m-1,m-1}|B_{1,1}, \ldots, B_{m-1,m-2}) = \prod_{i=1}^{m-1} \prod_{j=1}^{m-1} (\boldsymbol{M}^{+}_{i,j,*} - \boldsymbol{M}^{-}_{i,j,*})^{-1}, \end{aligned} \tag{11}$$

where $\boldsymbol{M}^{+}_{i,j,*}$ and $\boldsymbol{M}^{-}_{i,j,*}$ are updated before each subsequent element is drawn and conditioning on β is dropped for readability. Consequently, every draw from $h(\boldsymbol{B})$ is a eligible GPUC parameter matrix candidate and hence, (9) simplifies to:

$$p(\boldsymbol{M}|\beta) \approx \left(\frac{1}{N} \sum_{n=1}^{N} h(\boldsymbol{B}_n|\beta)^{-1} \right)^{-1}. \tag{12}$$

A comparison between a naive Monte Carlo approximation and a importance sampling approximation using both 20,000 draws for $m = 4$ (i.e. 9 parameters need to be drawn) is given in figure 1. Both approaches work for 4×4 matrices but the variance of the importance sampling version is already substantially lower. Additionally only 0.03% of all matrices drawn by the naive Monte Carlo simulation are eligible candidates for GPUC matrices. This proportion diminishes sharply, as m increases and hence, makes estimation via Monte Carlo infeasible. Once a sample is obtained, the prior is approximated as a function of β using a log-polynomial model of order k:

$$\widehat{log(P(\boldsymbol{M}|\beta))} = \sum_{i=0}^{k} \alpha_i \beta^i + \alpha_{k+1} log(\beta) \tag{13}$$

which serves as a very accurate approximation. Table 3 in the appendix displays estimates for $k = 5$ and $m = 3, \ldots, 20$.

5 MCMC algorithms for GPUC Matrices

Besides putting a (conditional) prior on $\boldsymbol{M}$ another necessity is to recompute the parameter matrix whenever a new value of $\widetilde{\beta}$ is proposed, because $\boldsymbol{M} \notin \mathcal{M}|\widetilde{\beta}$, i.e $\boldsymbol{M}$ is no longer an eligible GPUC parameter matrix. The following algorithm will

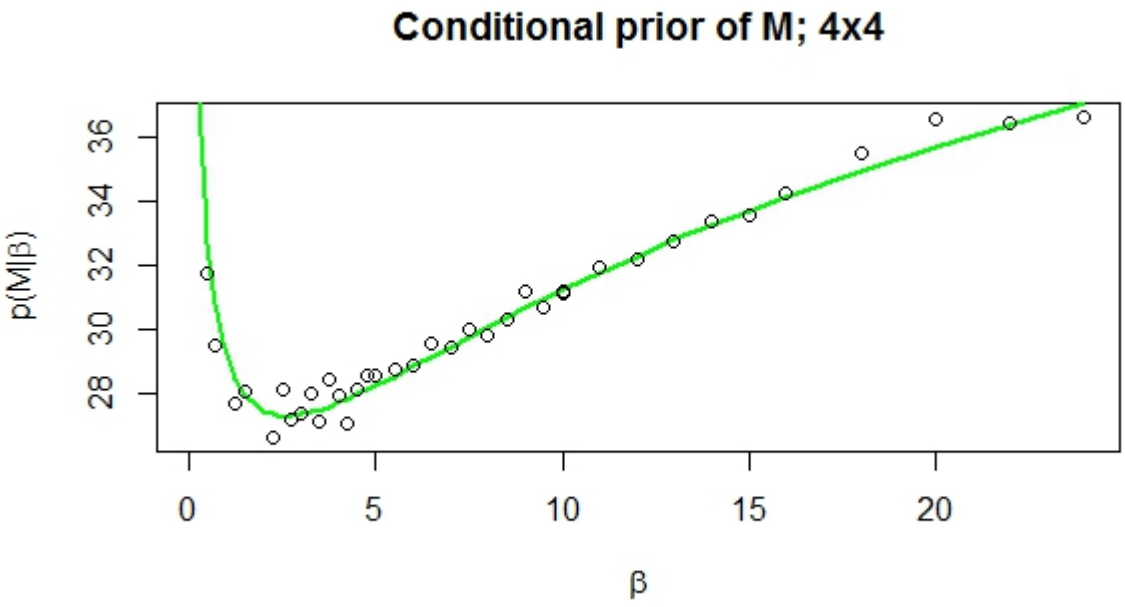

Figure 1: *Comparison of a naive Monte Carlo approximation (black dots) and the importance sampling approach (green line). 20,000 draws each for a negative binomial GPUC.*

thus be used to reshape the GPUC matrix $\boldsymbol{M}$:

Initialize $\boldsymbol{M}_c = \boldsymbol{M}$, $\boldsymbol{M}_r = \boldsymbol{M}$ and $\boldsymbol{M}_n = \boldsymbol{M}$; **while** $|\sum_{i=1}^{m} M_{n;i,j} - \alpha_j| \geq \varepsilon$ *for any* j *or* $|\sum_{i=j}^{m} M_{n;i,j} - \alpha_i| \geq \varepsilon$ *for any* i **do**

  Set $M_{r;i,\cdot} = M_{r;i,\cdot} \cdot \frac{\alpha_i(\beta_t)}{\alpha_i(\beta_{t-1})}$ for all i;

  Set $M_{c;\cdot,j} = M_{c;\cdot,j} \cdot \frac{\alpha_j(\beta_t)}{\alpha_j(\beta_{t-1})}$ for all j;

  Set $\boldsymbol{M}_n = \frac{\boldsymbol{M}_r + \boldsymbol{M}_c}{2}$;

  Set $\boldsymbol{M}_c = \boldsymbol{M}_n$ and $\boldsymbol{M}_r = \boldsymbol{M}_n$;

end

Set $\boldsymbol{M} = \boldsymbol{M}_n$

Algorithm 2: Reshaping GPUC matrices

To build a classical random walk Metropolis Hastings algorithm and use the posterior sample for parameter inference,two properties of GPUC parameter matrices are important:

Proposition 5.0.1. *Consider a GPUC parameter matrix* $\boldsymbol{M} \in \mathcal{M}_\beta$. *Further, set* $(i^*, j^*) = argmin_{i,j}\left(0.5 - \left|\frac{M_{i,j}}{\alpha_{max(i,j)}} - 0.5\right|\right)$. *Then, for two arbitrary Matrices* $\boldsymbol{G} \in \mathcal{M}_\beta$

and $\boldsymbol{H} \in \mathcal{M}_\beta$, $\boldsymbol{M}^* = \boldsymbol{M} + b(\boldsymbol{G} - \boldsymbol{H}) \in \mathcal{M}_\beta$, *if* $b \leq min\left(0.5 - \left|\frac{M_{i^*,j^*}}{\alpha_{max(i^*,j^*)}} - 0.5\right|, 0.5 - \left|\frac{\alpha_1 - g}{\sum_{i=2}^m \alpha_i} - 0.5\right|\right)$.

For the proof consider an arbitrary cell of $\boldsymbol{M}$, $M_{k,l} = g$ with $k \geq l$. Consequently no more than $\alpha_k - g$ can be additionally put in $M_{k,l}$ but by adding $b(G_{k,l} - H_{k,l})$ at most $b\alpha_k$ might be added and hence,

$$g + b\alpha_k \leq \alpha_k \Leftrightarrow b \leq 1 - \frac{g}{\alpha_k}.$$

On the other side, $M_{k,l}$ must not decreased by more than g if $k \neq 1$ or $l \neq 1$ or $\alpha_1 - \sum_{i=2}^m \alpha_i \leq 0$ and might be decreased at most by $b\alpha_k$ by adding $b(G_{k,l} - H_{k,l})$ and hence,

$$g - b\alpha_k \geq 0 \Leftrightarrow b \leq \frac{g}{\alpha_k}.$$

Accordingly, $b \leq 0.5 - \left|\frac{g}{\alpha_k} - 0.5\right|$.
If $k = 1$ and $l = 1$ and $\alpha_1 - \sum_{i=2}^m \alpha_i > 0$, then $M_{1,1}$ might be decreased by no more than $g - \alpha_1 + \sum_{i=2}^m \alpha_i$ but by adding $b(G_{1,1} - H_{1,1})$ at most $b\sum_{i=2}^m \alpha_i$ might be subtracted. Hence,

$$g - b\sum_{i=2}^m \alpha_i \geq \alpha_1 - \sum_{i=2}^m \alpha_i \Leftrightarrow b \leq \frac{g - \alpha_1}{\sum_{i=2}^m \alpha_i} + 1$$

On the other hand, $M_{1,1}$ must not increased to more than α_1 and can be increased by adding $b(\boldsymbol{G}_{1,1} - \boldsymbol{H}_{1,1})$ by no more than $\sum_{i=2}^m \alpha_i$. Thus,

$$g + b\sum_{i=2}^m \alpha_i \leq \alpha_1 \Leftrightarrow b \leq \frac{\alpha_1 - g}{\sum_{i=2}^m \alpha_i}.$$

Consequently, $b \leq 0.5 - \left|\frac{\alpha_1 - g}{\sum_{i=2}^m \alpha_i} - 0.5\right|$
The proof for $k \leq l$ follows analog and is omitted, here.
Proposition 5.0.1 states that it is always possible to find a value b to add the difference of two GPUC matrices to another matrix so that this sum is still an eligible GPUC matrix, a property that will be helpful in formulating a RW-MH algorithm.

Proposition 5.0.2. *Consider two matrices $\boldsymbol{M}_{m\times m} \in \mathcal{M}_\beta$ and $\boldsymbol{N}_{m\times m} \in \mathcal{M}_\beta$. Then for $\lambda \in [0,1]$, $\lambda\boldsymbol{M} + (1-\lambda)\boldsymbol{N} \in \mathcal{M}_\beta$.*

For a proof it is sufficient to assure that $\sum_{i=0}^{m-1} \lambda M_{ij} + (1-\lambda)N_{ij} = \sum_{i=0}^{m-1} \lambda M_{ij} + \sum_{i=0}^{m-1}(1-\lambda)N_{ij} = \alpha_j$. All other properties to fit definition (6) are fulfilled, by definition.

With this at hand, it is now possible to construct a random walk Metropolis Hastings algorithm the following way:

Initialize parameters $\boldsymbol{M^{(0)}}$ and $\beta^{(0)}$;
for *i in 1:N* **do**
  Propose $\widetilde{\beta}^{(i)} = \beta^{(i-1)} + \varepsilon$, with $\varepsilon \sim N(0,1)$;
  Recompute $\boldsymbol{M^{(i)}}$ using algorithm 2;
  Propose $\widetilde{\boldsymbol{M}}^{(i)} = \boldsymbol{M}^{(i)} + b(\rho_1 - \rho_2)$, with $\rho_1, \rho_2 \sim F(\boldsymbol{M})$ and $b = \frac{0.1}{ln(t)}$;
  If $\widetilde{\boldsymbol{M}}^{(i)} \in \mathcal{M}$, accept $\widetilde{\boldsymbol{M}}^{(i)}$ with probability $a = min\left(\frac{p(\widetilde{\boldsymbol{M}}^{(i)}, \widetilde{\beta}^{(i)}|u,v)}{p(\boldsymbol{M}^{(i-1)}, \beta^{(i-1)}|u,v)}, 1\right)$
end

Algorithm 3: Random Walk Metropolis Hasting for GPUC

In this algorithm, adding $b(\rho_1 - \rho_2)$ ensures the random walk property of the proposal. The decreasing scaling parameter b if helpful to ensure a relatively stable acceptance ratio, because $\widetilde{\boldsymbol{M}}^{(i)} \in \mathcal{M}$ holds for a large share of proposed matrices. On the down side a low (and even decreasing) scaling parameter b leads to poor mixing of the Markov chain and hence, slow convergence. To overcome this issue, an alternative, random blocking sampling scheme is proposed. This sampler was originated by Chib and Ramamurthy (2010) in the framework of estimating large DSGE models. Chib and Ramamurthy (2010) propose to divide the random walk Metropolis-Hastings (RW-MH) algorithm into two steps. The first step consists of randomly generating blocks of parameters whereas the second step is the classical RW-MH over the randomly blocked parameters. Both, the blocks and parameters are drawn in each iteration. In the setting of estimating DSGE models "genetaring random blocks is [...] straightforward and does not require comment", in the setting of this paper, however, the method of generating blocks is essential, due to the nature of the parameter space. The randomized blocking random walk Metropolis Hastings (RBRW-MH) algorithm used here can

be summarized as follows:

Initialize $\boldsymbol{R} = \mathbf{1}_{m\times m}$; $\boldsymbol{M}^{(0)}$ and $\beta^{(0)}$;
for *t in 1:N* **do**
  Propose $\beta^{(t)} = \beta^{(t-1)} + \varepsilon$, with $\varepsilon \sim N(0,1)$;
  Recompute $\boldsymbol{M}^{(t)}$ using algorithm 2;
  Accept $\left(\boldsymbol{M}^{(t)}, \beta^{(t)}\right)$ with probability $\alpha = min\left(\frac{p(\boldsymbol{M}^{(t)},\beta^{(t)}|u,v)}{p(\boldsymbol{M}^{(t-1)},\beta^{(t-1)}|u,v)}, 1\right)$;
  Set $\widetilde{\boldsymbol{M}}^{(0)} = \boldsymbol{M}^{(t)}$;
  for *r in 1:*$(m^2/4)$ **do**
    Draw i and j from $\{1,\ldots,m\}$, with $R_{i,j} = 1$;
    Draw k from $\{1,\ldots,m\} \setminus \{i, s : R_{i,s} = 0\}$;
    Draw l from $\{1,\ldots,m\} \setminus \{j, t : R_{t,j} = 0\}$;
    Set $R_{i,j} = R_{k,l} = R_{i,l} = R_{k,j} = 0$;
    Draw $\varrho \sim N(0,q)$, $q = \frac{\gamma \alpha_{max(i,j,k,l)}}{ln(t)}$;
    Set $\widetilde{\boldsymbol{M}}^{(r)} = \widetilde{\boldsymbol{M}}^{(r-1)}$;
    Compute $\widetilde{M}^{(r)}_{i,j} = \widetilde{M}^{(r)}_{i,j} + \varrho$, $\widetilde{M}^{(r)}_{i,l} = \widetilde{M}^{(r)}_{i,l} - \varrho$, $\widetilde{M}^{(r)}_{k,j} = \widetilde{M}^{(r)}_{k,j} - \varrho$, $\widetilde{M}^{(r)}_{k,l} = \widetilde{M}^{(r)}_{k,l} + \varrho$;
    Accept $(\widetilde{\boldsymbol{M}}^{(r)})$ with probability $\alpha = min\left(\frac{p(\widetilde{\boldsymbol{M}}^{(r)},\beta^{(t)}|u,v)}{p(\widetilde{\boldsymbol{M}}^{(r-1)},\beta^{(t)}|u,v)}, 1\right)$;
  end
  Set $\boldsymbol{M}^{(t)} = \widetilde{\boldsymbol{M}}^{(m^2/4)}$;
  Set $\boldsymbol{R} = \mathbf{1}_{m\times m}$;
end

Algorithm 4: Randomized Blocking Random Walk Metropolis Hastings for Generalized Partition of Unity Copulas.

The algorithm proposes a new value for β at the beginning of each iteration. Once β is proposed, the parameter matrix needs to be recomputed in order to have correct row and column wise sums. After accepting or rejecting the new value of β, the algorithm selects four elements from $\boldsymbol{M}$ according to Table 1 and adds ϱ to the elements (i,j) and (k,l) and subtracts ϱ from elements (i,l) and (k,j). If ϱ is small in absolute value the probability of proposing a matrix $\widetilde{\boldsymbol{M}}^{(r)} \in \mathcal{M}$ is sufficiently high to obtain a well mixing Markov Chain, due to proposition 5.0.1.

$$
\mathrm{M}=\begin{bmatrix}
M_{1,1} & \dots & M_{1,j} & \dots & M_{1,l} & \dots & M_{1,m} \\
\vdots & & \vdots & & \vdots & & \vdots \\
M_{i,1} & \dots & M_{i,j} & \dots & M_{i,l} & \dots & M_{i,m} \\
\vdots & & \vdots & & \vdots & & \vdots \\
M_{k,1} & \dots & M_{k,j} & \dots & M_{k,l} & \dots & M_{k,m} \\
\vdots & & \vdots & & \vdots & & \vdots \\
M_{m,1} & \dots & M_{m,j} & \dots & M_{m,l} & \dots & M_{m,m}
\end{bmatrix}
$$

Table 1: *Choosing four elements of M*

Once a new parameter matrix $\widetilde{\boldsymbol{M}}^{(r)}$ is proposed, the computing the likelihood $p(u, v|\widetilde{\boldsymbol{M}}^{(r)}, \beta^{(t)})$ boils down to calculating:

$$
\begin{aligned}
p\left(u, v|\widetilde{\boldsymbol{M}}^{(r)}, \beta^{(t)}\right) &= \prod_{s=1}^{N} c\left(u_s, v_s|\widetilde{\boldsymbol{M}}^{(r)}, \beta^{(t)}\right) \\
&= \prod_{s=1}^{N} c(u_s, v_s|\widetilde{\boldsymbol{M}}^{(r-1)}, \beta^{t}) \quad +\varrho\left(\frac{\phi_{i-1}(u_s)\phi_{j-1}(v_s)}{\alpha_{i-1}\alpha_{j-1}}\right) + \varrho\left(\frac{\phi_{k-1}(u_s)\phi_{l-1}(v_s)}{\alpha_{k-1}\alpha_{l-1}}\right) \\
&\qquad\qquad -\varrho\left(\frac{\phi_{i-1}(u_s)\phi_{l-1}(v_s)}{\alpha_{i-1}\alpha_{l-1}}\right) - \varrho\left(\frac{\phi_{k-1}(u_s)\phi_{j-1}(v_s)}{\alpha_{k-1}\alpha_{j-1}}\right)
\end{aligned}
$$

and hence, the acceptance probability α can be computed very efficiently. In fact, a single iteration of the outer loop of algorithm 4 is of the same complexity as one iteration of the RW-MH sampler in Algorithm 3: $\mathcal{O}(m^2 n)$

6 Simulation studies

This section compares both proposed samplers by using simulated data sets. The first data set is generated from a negative binomial GPUC with $\beta = 2$ and a diagonal matrix $\boldsymbol{M}$. In this setting reduced form and normal form of the copula coincide and hence, the dimension of $\boldsymbol{M}$ should not play a major role regarding the ability to adequately mimic the data. Both samplers are compared estimating a 4×4 matrix and a 10×10 matrix and are run for $5,000$ iterations with the first

1,000 being discarded as burn-in sample. The second and third data sets are generated by Bernstein Copulas with parameter matrices

$$\boldsymbol{M}_1 = \begin{bmatrix} 0 & 0 & 0 & 0.25 \\ 0 & 0 & 0.25 & 0 \\ 0 & 0.25 & 0 & 0 \\ 0.25 & 0 & 0 & 0 \end{bmatrix}, \boldsymbol{M}_2 = \begin{bmatrix} 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \end{bmatrix}$$

and both samplers are run for 20,000 iterations with the first 10,000 observations being discarded as burn-in sample. Table 2 shows the computing times (CT) in seconds and efficient sample sizes (ESS) of both samplers; the values in parentheses show the effective sample size, given $\beta = \hat{\beta} = \frac{1}{N}\sum_{i=1}^{N}\beta_i$. Point estimates are computed by averaging over the posterior sample (Bernstein Copula) and first averaging over β and then, computing the average parameter matrix $\boldsymbol{M}$ given $\hat{\beta}$ (negative Binomial GPUC) according to proposition 5.0.2. Obviously, the random blocking Metropolis-Hastings sampler outperforms the classical Metropolis-Hastings algorithm for both copulas and matrix sizes. The time, necessary to compute a single independent observation of the posterior [3] is much lower for the RBRW-MH algorithm (NB-GPUC: 31s and 78s; Bernstein: 3 and 49s) compared to the RW-MH algorithm (NB-GPUC: 144s and 971s; Bernstein Copula: 11s and 200s). By investigating both, the likelihood and trace plots for the Bernstein Copula (see figure 2) it is clear that the Markov chain generated by the classic Metropolis Hastings algorithm has not even converged, yet, whereas the RBRW-MH algorithm already converged after roughly 200 (4×4) and 1,000 (10×10) iterations, respectively.

[3] The efficient sample size (ESS) is a measure for the number of independent draws from the posterior; see eg. Gelman et al. (2004). Computing ESS was done using the *coda* package in R.

		ESS	**CT**	$lnL(\widehat{\boldsymbol{M}}, \hat{\beta})$
NB GPUC 4×4	RW-MH	48 (4)	6,926	689,1
	RBRW-MH	163 (133)	5,120	691,9
NB GPUC 10×10	RW-MH	9 (5)	8,747	675,1
	RBRW-MH	67 (62)	5,258	690,3
Bernstein Copula 4×4	RW-MH	44	500	370.4
	RBRW-MH	240	641	403.3
Bernstein Copula 10×10	RW-MH	7	1,401	2795.2
	RBRW-MH	122	6,099	3140.7

Table 2: *Effective Sample Size, computing time and log likelihood of the proposed MCMC algorithms.*

7 Concluding Remarks

This paper proposes two MCMC algorithms to estimate Generalized Partition of Unity Copulas beginning with an algorithm to sample the respective parameter matrices. It was shown that a uninformative prior distribution of the parameter matrix given other parameters of the GPUC can be computed using importance sampling. Two simulation studies compared the algorithms and verified that the proposed random blocking random walk Metropolis-Hastings algorithm has promising properties, ie. fast convergence and low autocorrelation. Hence, this framework offers a way to incorporate GPUC into various types of models.

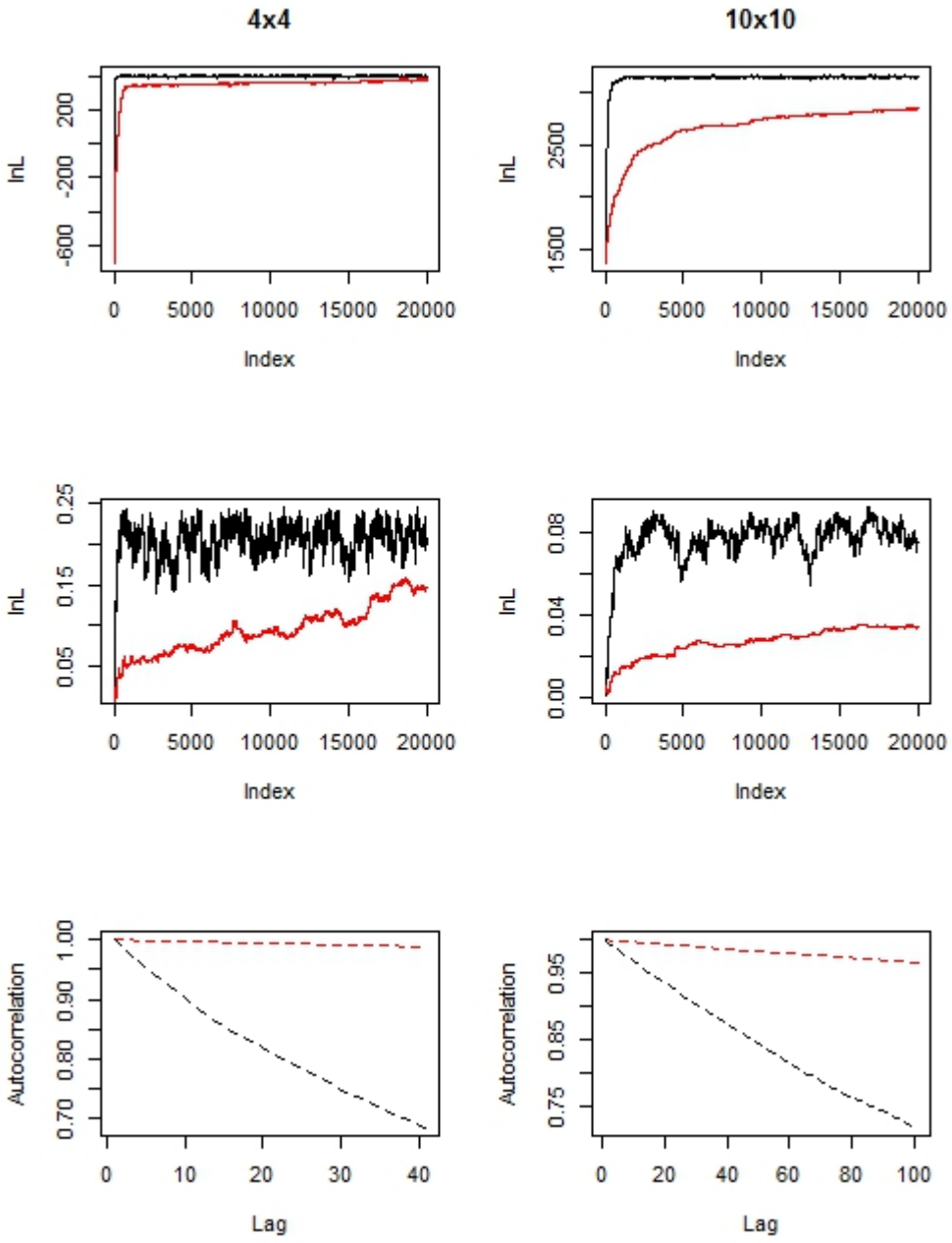

Figure 2: *Comparison of the RBRW-MH and the RW-MH algorithm when estimating a Bernstein Copula. The left column shows results for a simulated data set with* $\boldsymbol{M} = \boldsymbol{M_1}$ *and the right column with* $\boldsymbol{M} = \boldsymbol{M_2}$*. The black lines represent the RBRW-MH and the red lines represent the RW-MH algorithm.*

8 Appendix

8.1 Log-polynomial fit of the prior

m	$const.$	β	β^2	β^3	β^4	β^5	$log(beta)$
3	-8.168	-2.700	0.264	-0.017	0.001	-0.00001	3.661
4	-24.396	-5.269	0.475	-0.030	0.001	-0.00001	8.523
5	-52.010	-8.234	0.690	-0.042	0.001	-0.00002	15.414
6	-92.579	-11.511	0.908	-0.054	0.002	-0.00002	24.353
7	-147.882	-14.515	1.029	-0.057	0.002	-0.00002	35.040
8	-217.445	-18.972	1.360	-0.077	0.002	-0.00003	48.653
9	-307.830	-20.788	1.363	-0.088	0.004	-0.0001	61.793
10	-416.362	-24.294	1.089	-0.039	0.001	-0.00001	80.206
11	-543.687	-32.864	1.984	-0.101	0.003	-0.00003	102.120
12	-709.766	-29.688	0.539	0.052	-0.004	0.0001	118.946
13	-891.356	-35.922	0.527	0.074	-0.005	0.0001	144.568
14	$-1,123.319$	-24.221	-2.140	0.317	-0.015	0.0002	153.370
15	-1,337.284	-67.008	5.310	-0.426	0.019	-0.0003	208.712
16	-1,658.292	-42.240	1.154	-0.091	0.006	-0.0001	210.521
17	-1,943.941	-73.886	4.443	-0.301	0.013	-0.0002	265.396
18	-2,333.162	-52.893	-2.322	0.467	-0.023	0.0003	289.127
19	-2,750.608	-50.188	-1.457	0.220	-0.008	0.0001	311.079
20	-3,158.849	-79.790	1.214	0.115	-0.006	0.0001	368.077

Table 3: *Estimates of the log-polynomial model for a conditionally flat prior of the negative binomial GPUC.*

8.2 Beta-Mixture representation of negative Binomial GPUC and Bernstein Copula.

Consider a random variable $X \sim Beta(p, q)$, then the density of X is given by

$$f_X^{(p,q)}(x) = \frac{(p+q-1)!}{(p+1)!(q+1)!} x^{p-1}(1-x)^{q-1}.$$

The density function of a negative binomial GPUC with parameter matrix $\boldsymbol{M}$ and shape parameter β is given by

$$\begin{aligned} c(u,v) &= \sum_i \sum_j \frac{M_{i,j}}{\alpha_i \alpha_j} \binom{\beta+i-1}{i} \binom{\beta+j-1}{j} (1-u)^\beta (1-v)^\beta u^i v^j \\ &= \sum_i \sum_j \frac{M_{i,j}}{\alpha_i \alpha_j} \frac{(\beta+i-1)!}{i!(\beta-1)!} \frac{(\beta+j-1)!}{j!(\beta-1)!} (1-u)^\beta (1-v)^\beta u^i v^j. \end{aligned}$$

Now, for the negative Binomial GPUC:

$$\alpha_i = \int_0^1 \binom{\beta+i-1}{i} u^i (1-u)^\beta du = \frac{\beta}{(\beta+i)(\beta+i+1)}$$

and hence,

$$\begin{aligned} c(u,v) &= \sum_i \sum_j \Bigg(M_{i,j} \frac{(\beta+i)(\beta+i+1)(\beta+j)(\beta+j+1)}{\beta^2} \frac{(\beta+i-1)!}{i!(\beta-1)!} \frac{(\beta+j-1)!}{j!(\beta-1)!} \\ &\qquad (1-u)^\beta (1-v)^\beta u^i v^j \Bigg) \\ &= \sum_i \sum_j M_{ij} \frac{(\beta+i+1)!(\beta+j+1)!}{i!\beta j!\beta} (1-u)^\beta (1-v)^\beta u^i v^i \\ &= \sum_i \sum_j M_{ij} f^{(i+1,\beta+1)}(u) f^{(j+1,\beta+1)}(v) \quad \square \end{aligned}$$

The density function of a Bernstein Copula with parameter matrix $\boldsymbol{M}_{(m \times m)}$ and $\alpha_i = \frac{1}{m}$ is given by:

$$\begin{aligned} c(u,v) &= \sum_i \sum_j m^2 M_{ij} \binom{m-1}{i} \binom{m-1}{j} u^i (1-u)^{m-i-1} v^j (1-v)^{m-j-1} \\ &= \sum_i \sum_j M_{ij} \frac{m!}{i!(m-i-1)!} \frac{m!}{j!(m-j-1)!} u^i (1-u)^{m-i-1} v^j (1-v)^{m-j-1} \\ &= \sum_i \sum_j M_{ij} f^{(i+1,m-i)}(u) f^{(j+1,m-j)}(v) \quad \square \end{aligned}$$

References

Baker, R. (2008). An order-statistics-based method for constructing multivariate distributions with fixed marginals. *Journal of Multivariate Analysis 99*(10), 2312–2327.

Bedford, T. and R. M. Cooke (2002). Vines–a new graphical model for dependent random variables. *The Annals of Statistics 30*(4), 1031–1068.

Chib, S. and S. Ramamurthy (2010). Tailored randomized block mcmc methods with application to dsge models. *Journal of Econometrics 155*(1), 19 – 38.

Dou, X., S. Kuriki, G. D. Lin, and D. Richards (2016). Em algorithms for estimating the bernstein copula. *Computational Statistics & Data Analysis 93*, 228–245.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin (2004). *Bayesian data analysis* (2. ed. ed.). Texts in statistical science. Boca Raton, Fla.: Chapman & Hall.

Greenberg, E. (2008). *Introduction to Bayesian econometrics*. Cambridge: Cambridge University Press.

Nelsen, R. B. (2006). *An Introduction to Copulas* (Second Edition ed.). Springer Series in Statistics. New York, NY: Springer Science+Business Media Inc.

Pfeifer, D., A. Mändle, and O. Ragulina (2017). Data driven partition-of-unity copulas with applications to risk management.

Pfeifer, D., H. A. Tsatedem, A. Mändle, and C. Girschig (2016). New copulas based on general partitions-of-unity and their applications to risk management. *Dependence Modeling 4*(1), 225.

Savu, C. and M. Trede (2010). Hierarchies of archimedean copulas. *Quantitative Finance 10*(3), 295–304.