A Petrov Galerkin projection for copula density estimation

D. Uhlig, R. Unger

Preprint 2013-07

Preprintreihe der Fakultät für Mathematik
ISSN 1614-8835

Impressum:
Herausgeber:
Der Dekan der
Fakultät für Mathematik
an der Technischen Universität Chemnitz
Sitz:
Reichenhainer Strae 39
09126 Chemnitz
Postanschrift:
09107 Chemnitz
Telefon: (0371) 531-22000
Telefax: (0371) 531-22009
E-Mail: dekanat@mathematik.tu-chemnitz.de
Internet:
http://www.tu-chemnitz.de/mathematik/
ISSN 1614-8835 (Print)

A Petrov Galerkin projection for copula density estimation

July 3, 2013

Dana Uhlig
Chemnitz University of Technology
Department of Mathematics
09107 Chemnitz, Germany
dana.uhlig@mathematik.tu-chemnitz.de

Roman Unger
Chemnitz University of Technology
Department of Mathematics
09107 Chemnitz, Germany
roman.unger@mathematik.tu-chemnitz.de

Abstract

The reconstruction of the dependence structure of two or more random variables ($d \geq 2$) is a big issue in finance and many other applications. Looking at samples of the random vector, neither the common distribution nor the copula itself are observable. So the identification of the copula C or the copula density $\mathrm{c}(u_1, \ldots, u_d) = \frac{\partial^d \mathrm{C}}{\partial u_1 \ldots \partial u_d}$ can be treated as an inverse problem. In the statistical literature usually kernel estimators or penalized maximum likelihood estimators are considered for the non-parametric estimation of the copula density c from given samples of the random vector. Even though the copula C itself is unobservable we can treat the empirical copula as a noisy representation, since it is well known that the empirical copula converges for large samples to the copula and solve the $d-$dimensional linear integral equation $\mathrm{C}(u) = \int_0^u \mathrm{c}(s)ds$ for determining the copula density c. We present a Petrov-Galerkin projection for the numerical computation of the linear integral equation and discuss the assembling algorithm of the non-sparce matrices and vectors. Furthermore we analyze the stability of the discretized linear equation.

1 Introduction

The common distribution function F contains the complete information about the behaviour of the random vector $\mathbf{X} = [X_1, \ldots, X_d]^T$. To analyse the dependence structure it is useful to separate it from the marginals in particular if $\mathbf{X}$ isn't Gaussian distributed. The tool is the copula and we demonstrate the splitting for the easiest case of continuous marginals F_i. Then it is well known that the random variables $U_i = F_i(X_i)$ are uniformly distributed and we obtain the decomposition

$$\begin{aligned} F(\mathbf{x}) &= P(X_1 \leq x_1, ..., X_d \leq x_d) = P(U_1 \leq F_1(x_1), \ldots, U_d \leq F_d(x_d)) \\ &= C(F_1(x_1), \ldots, F_d(x_d)) \,. \end{aligned}$$

Hence a copula is in principle the common distribution of a random vector $\mathbf{U} = [U_1, \ldots, U_d]^T$ with uniformly distributed components $U_i \sim U[0,1]$ and contains the complete information about the dependence structure. For a detailed introduction to copulas and their properties see [Nel06] or [MFE10, Chapter 5]. There is a large field of applications in finance, actuarial mathematics, survival analysis and climate research, just to mention a few. Sklar's Theorem, see for example [Nel06, Theorem 2.10.9], ensures the existence of a copula.

Proposition 1.1. *Sklar's Theorem (1959)*

1. *Let F be an $d-$dimensional distribution function with margins $F_1, \ldots, F_d$. Then there exists an $d-$dimensional copula C with*

$$F(x_1, \ldots, x_d) = C(F_1(x_1), \ldots, F_d(x_d)) \quad \forall \mathbf{x} \in \mathbb{R}^d. \tag{1}$$

   C is uniquely determined on $range(F_1) \times \ldots \times range(F_d)$, that is for continuous margins $F_1, \ldots, F_d$ C is unique.

2. *If C is $d-$dimensional copula and $F_1, \ldots, F_d$ are distribution functions, then the function F in (1) is an $d-$dimensional distribution with margins $F_1, \ldots, F_d$.*

We will give a brief overview of the properties which are important for the estimation. A copula is smooth in the following sense (see [MS12, Lemma 1.2]).

Lemma 1.2. *Smoothness of a copula*

1. *Lipschitz continuity: for every* $\mathbf{u} = (u_1, \ldots, u_d), \mathbf{v} = (v_1, \ldots, v_d) \in [0,1]^d$ *it holds*
$$|C(\mathbf{u}) - C(\mathbf{v})| \leq \sum_{i=1}^{d} |u_i - v_i|$$

2. *partial derivatives: For fixed* $(u_1, \ldots, u_{k-1}, u_{k+1}, \ldots, u_d) \in [0,1]^{d-1}$ $(k = 1, \ldots, d)$ *the partial derivative* $u_k \mapsto \frac{\partial}{\partial u_k} C(u_1, \ldots, u_d)$ *exists (Lebesgue) almost everywhere on* $[0,1]$ *and* $0 \leq \frac{\partial}{\partial u_k} C(u_1, \ldots, u_d) \leq 1$.

If the copula C is absolutely continuous there is a almost everywhere unique copula density $\mathrm{c} : [0,1]^d \to [0, \infty)$ such that

$$\mathrm{C}(u) = \int_0^u \mathrm{c}(s) ds \tag{2}$$

and then it yields

$$\mathrm{c}(u_1, \ldots, u_d) = \frac{\partial^d \mathrm{C}}{\partial u_1 \ldots \partial u_d} \quad u_1, \ldots, u_d \in (0,1) \tag{3}$$

almost everywhere (see [MS12]). For non absolutely continuous copulas there is a decomposition

$$\mathrm{C} = A_{\mathrm{C}} + S_{\mathrm{C}} \tag{4}$$

in an absolutely continuous component

$$A_{\mathrm{C}}(u_1, \ldots, u_d) = \int_0^{u_1} \cdots \int_0^{u_d} \frac{\partial^d \mathrm{C}}{\partial s_1 \cdots \partial s_d} ds_1 \cdots ds_d$$

and a singular component S_{C} such that the support of S_{C} has Lebesgue measure zero (see [Nel06, Chapter 2.4]).

1.1 Inverse Problem

The copula contains the complete information about the dependence structure of the random vector $\mathbf{X} = [X_1, \ldots, X_d]^T$. In many applications the slopes at the corners are of special interest. Therefore the reconstruction of

the copula density (2) is a big issue. Of course (3) exists only for absolutely continuous copulas. In this case we consider the inverse problem

$$\mathrm{C}(u) = \int_0^u \mathrm{c}(s)ds \quad \forall u \in \Omega := [0,1]^d \tag{5}$$

that means we want to determine the density from a given copula. Solving the linear integral equation (5) requires an adequate knowledge of the copula C which is in general not observable. However, in the statistical framework we observe T samples $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_T$ with $\mathbf{X}_j = [X_{1j}, \ldots, X_{dj}]^T$ of the random vector $\mathbf{X} = [X_1, \ldots, X_d]^T$ and want to compute the density of the copula in order to get information about the dependence structure of the random vector $\mathbf{X}$. For this reason we have to transform the samples of the $k-$th component $U_{kj} = F_k(X_{kj})$ for $k = 1, \ldots, d$ by the margins F_k, which are usually unknown. In this case we compute the pseudo observations $\hat{\mathbf{U}}_1, \hat{\mathbf{U}}_2, \ldots, \hat{\mathbf{U}}_T$ using the empirical distribution functions

$$\hat{F}_k(x) = \frac{1}{T} \sum_{j=1}^{T} \mathbb{1}_{X_{kj} \leq x} \tag{6}$$

such that $\hat{\mathbf{U}}_j = (\hat{F}_1(X_{1j}), \ldots, \hat{F}_d(X_{dj}))^T$ and compute the empirical copula

$$\hat{\mathrm{C}}(u) = \frac{1}{T} \sum_{j=1}^{T} \mathbb{1}_{\hat{\mathbf{U}}_j \leq u} = \frac{1}{T} \sum_{j=1}^{T} \prod_{k=1}^{d} \mathbb{1}_{\hat{U}_{kj} \leq u_k} \,. \tag{7}$$

It is well known that the empirical copula uniformly converges to the copula (see [Deh80])

$$\max_{u \in [0,1]^d} \left| \mathrm{C}(u) - \hat{\mathrm{C}}(u) \right| = \mathcal{O}\left(\frac{(\log\log T)^{\frac{1}{2}}}{T^{\frac{1}{2}}} \right) \quad \text{a.s.} \tag{8}$$

Therefore we treat the empirical copula as a noisy representation of the unobservable copula $\mathrm{C}^\delta = \hat{\mathrm{C}}$ and solve

$$\int_0^u \mathrm{c}(s)ds = \mathrm{C}^\delta(u) \quad \forall u \in \Omega = [0,1]^d \tag{9}$$

instead of equation (5) if the copula C is not known.

2 Solving the integral equation

2.1 Petrov-Galerkin projection

The computation of the copula density c from a given copula C is in principal a numerical differentiation which is a well known ill-posed inverse problem. For solving (5) we propose the following discretization. We choose an ansatz space V_h with base $\Phi := [\phi_1, \phi_2 \cdots \phi_N]$ and test functions $\psi_i \in \tilde{V_h}$, set

$$\mathrm{c}_h(s) := \sum_{j=1}^{N} c_j \phi_j(s) \tag{10}$$

insert this in (5), multiply with the test function and integrate over Ω, which leads to

$$\int_\Omega \int_0^u \sum_{j=1}^{N} c_j \phi_j(s) ds \psi_i(u) du = \int_\Omega \mathrm{C}(u)\psi_i(u)du \quad \forall \psi_i \in \tilde{V_h} \tag{11}$$

which is the Petrov-Galerkin projection of (5).

Because of the linearity of the integral we can transform (11) over

$$\sum_{j=1}^{N} c_j \int_\Omega \int_0^u \phi_j(s) ds \psi_i(u) du = \int_\Omega \mathrm{C}(u)\psi_i(u)du \quad \forall \psi_i \in \tilde{V_h} \tag{12}$$

to a system of linear equations

$$Kc = C \qquad K \in \mathbb{R}^{N\times N} \quad c, C \in \mathbb{R}^N \tag{13}$$

where the system matrix K, and right hand side C is defined by

$$\begin{bmatrix} K_{ij} := \\ \\ \int_\Omega \int_0^u \phi_j(s) ds \psi_i(u) du \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} C_1 \\ \vdots \\ C_i := \int_\Omega \mathrm{C}(u)\psi_i(u)du \\ \vdots \\ C_N \end{bmatrix}$$

No we can choose ansatz functions ϕ_j by discretising the domain Ω in n^d intervals, squares, cubes or hypercubes (depending on dimension d) called e_j with

$$\Omega = \bigcup_{i=1}^{N} e_i \quad \text{and} \quad e_i \cap e_j = \emptyset \quad \text{if} \quad i \neq j \tag{14}$$

and set

$$\phi_j(u) := \begin{cases} 1 & u \in e_j \\ 0 & \text{otherwise}\,. \end{cases} \tag{15}$$

We enumerate the $N = n^d$ elements $e_1, \ldots, e_N$ spanning $\Omega = [0,1]^d$ in a special order. Actually, we start discretizing $[0,1]$ in the first elements $e_1, \ldots, e_n$. Then we discretize the second dimension such that we additionally obtain $e_{n+1}, \ldots, e_{n^2}$ following the third dimension and so on. For example in the case $d = 2$ this leads to a grid of squares numbered by $1, \cdots, n$ in the first row, $n+1, \cdots, 2n$ in the second and so on.

Furthermore we choose as test functions ψ_i the integrated ansatz functions

$$\psi_i(u) := \int_0^u \phi_i(s) ds \tag{16}$$

such that the system (13) becomes symmetric.

Note that the proposed ansatz works also for non absolutely continuous copulas. Since equation (4) we compute in this case the copula density of the absolutely continuous part superposed by a dirac distribution of the singular component S_{C}.

2.2 Structure of the system matrix

The special choice of the ansatz and test functions ψ_i leads to a particular structure of the system matrix K shown in the figure 1. Therefore it is not needed to assemble the whole $n^d \times n^d$ system matrix, which is an essential task. In contrast to finite element discretizations with sparse matrices the system (13) is not sparse and has the system size $N = n^d$. For moderate values of n and small values of d it can be assembled numerically like an arbitrary finite element system. But the assembling, storage and solving becomes impossible for ususal discretizations and dimensions. As example we consider the case $n = 80$ and $d = 3$, we have $N = 512000$. The storage

for the system matrix is then $(N^2 * 8/1024^3)$ approximatly 2000 Gigabyte, with symmetrie still 1 Terabyte and the computing times for assembling and solving such a system will become enourmous. There is an Kronecker product factorization of the matrix K which will developed in the following section.

We call the coordinates $b^i = [b_1^i, b_2^i, \cdots, b_d^i]^T$ of the lowest corner of element $e_i \in \Omega$ with $i \in \{1, 2, \ldots, n^d\}$ and they are defined component wise by $b_k^i = \min_{u \in e_i}(u_k)$ and use the tensor product structure of ψ_i, with $h = \frac{1}{n}$ and the auxilary function

$$\xi(t, b) = \begin{cases} 0 & t < b \\ (t - b) & b \leq t \leq b + h \\ h & t > b + h\,. \end{cases}$$

Lemma 2.1. *The integrated ansatz functions (16) fullfils*

$$\psi_i(u) = \prod_{k=1}^{d} \xi(u_k, b_k^i) \quad \forall i = 1, \ldots, N\,. \tag{17}$$

Furthermore, it holds

$$\psi_i(u) = 0 \quad \forall u \in e_j \quad \textit{with} \quad j < i \tag{18}$$

for arbitrary dimensions d.

Proof. Formula (17) is a direct consequence of Fubini's theorem and (18) follows directly from (17) since for at least one index k there is $b_k^j < b_k^i$. □

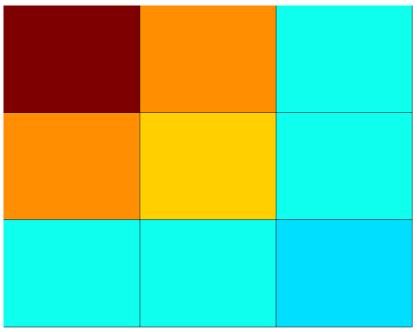
Lemma 2.2.

$$K_{i,j} = \int_{[0,1]^d} \psi_i(u)\psi_j(u)du = \prod_{k=1}^{d} \tilde{K}_{i,j}^k \tag{19}$$

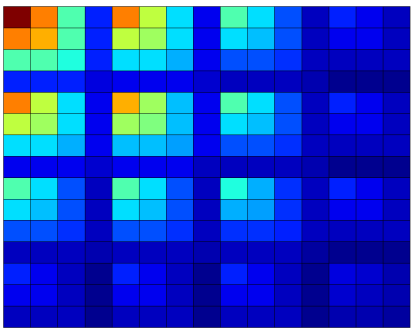
with

$$\tilde{K}_{i,j}^k = \int_{[0,1]} \xi(u, b_k^i)\xi(u, b_k^j)du = h^2\left(1 - \max\left\{b_k^i, b_k^j\right\}\right) - h^3 \begin{cases} \frac{2}{3}, & \textit{if } b_k^i = b_k^j \\ \frac{1}{2}, & \textit{if } b_k^i \neq b_k^j \end{cases} \tag{20}$$

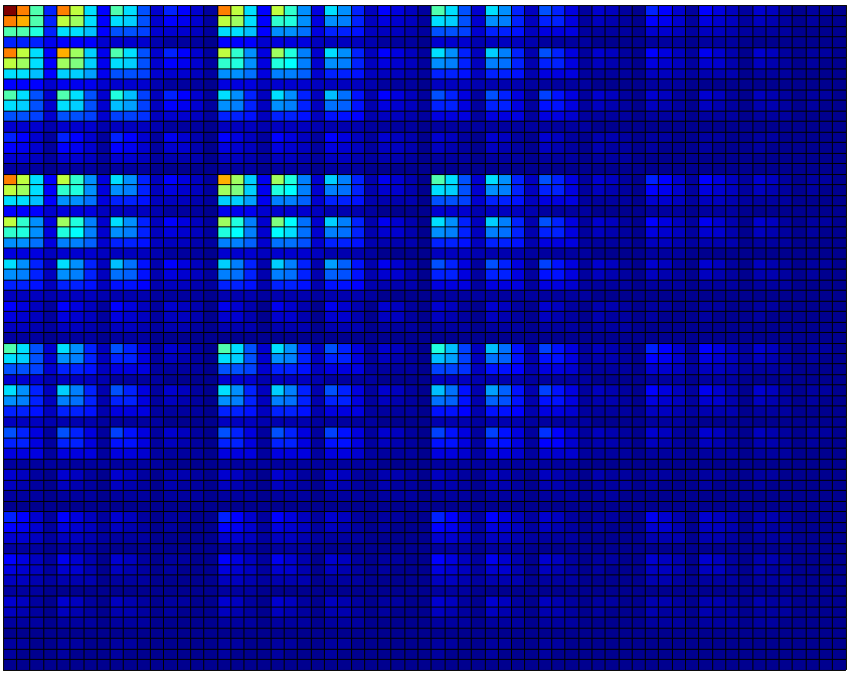
Using lemma 2.2 we only need to know the coordinates of the elements e_i and e_j for the computation of the elements $K_{i,j}$. Lemma 2.3 specifies the mapping from the element number to the coordinates and vice versa.



(a) System matrix for $d = 1$



(b) System matrix for $d = 2$



(c) System matrix for $d = 3$

Figure 1: Matrixplots of the system matrix K for $n = 4$ and different dimensions d.

Lemma 2.3. *For the element number i it yields*

$$i = 1 + \sum_{k=1}^{d} b_k^i n^k \tag{21}$$

and

$$b_k^i = h \left(\left\lceil \frac{i - \sum_{j=k+1}^{d} b_j^i n^j}{n^{k-1}} \right\rceil - 1 \right), \quad k = d, d-1, \ldots, 1 \tag{22}$$

If the dimension of the problem is increased from d to $d+1$ the number of elements also increases from n^d to n^{d+1}. For a better distinction of the coordinates we write ${}^{(d)}b^i$ and ${}^{(d+1)}b^i$ respectively.

Corollary 2.4. *It yields*

1. ${}^{(d+1)}b_k^i = {}^{(d)}b_k^i$ *for all* $i = 1, \ldots, n^d$ *and* $k = 1, \ldots, d$
2. ${}^{(d+1)}b_{d+1}^i = (j-1)h$ *for all* $i = (j-1)n^d + 1, \ldots, jn^d$ *with* $j = 1, \ldots, n$
3. ${}^{(d+1)}b_k^{i+jn^d} = {}^{(d)}b_k^i$ *for all* $k = 1, \ldots, d$, $i = 1, \ldots, n^d$ *and* $j = 0, \ldots, n-1$

Having regard to different dimensions we also write for the $n^d \times n^d$ system matrix K in the $d-$dimensional case ${}^{(d)}K$

$${}^{(d)}K = \begin{bmatrix} {}_{1,1}B & {}_{1,2}B & \cdots & {}_{1,n}B \\ \vdots & & & \vdots \\ {}_{n,1}B & {}_{n,2}B & \cdots & {}_{n,n}B \end{bmatrix} \tag{23}$$

and consider their n^2 block matrices ${}_{l,q}B$ of type $n^{d-1} \times n^{d-1}$.

Proposition 2.5. *The system matrix for the $(d+1)-$dimensional case can be extracted from the one and $d-$dimensional system matrices.*

$${}^{(d+1)}K = {}^{(1)}K \otimes {}^{(d)}K \tag{24}$$

Proof. Following corollary 2.4 the $(d+1)-$th coordinate of the element with number $(n^d(l-1)+r)$ is for all $r = 1, \ldots, n^d$ independent of r and it yields ${}^{(d+1)}b_{d+1}^{n^d(l-1)+r} = (l-1)h$. Likewise, the other first d coordinates of

the $(n^d(l-1)+r)-$th elements can be reduced to the $d-$dimensional case, since ${}^{(d+1)}b_k^{r+(l-1)n^d} = {}^{(d)}b_k^r$ for all $l = 1, \ldots, n$. Now considering the $(l,q)-$th block of ${}^{(d+1)}K$ using lemma 2.2 we compute

$$
\begin{aligned}
{}_{l,q}B_{r,s} = {}^{(d+1)}K_{n^d(l-1)+r,n^d(q-1)+s} &= \int\limits_{[0,1]^{d+1}} \psi_{n^d(l-1)+r}(u)\psi_{n^d(q-1)+s}(u)du \\
&= \prod_{k=1}^{d+1} \tilde{K}^k_{n^d(l-1)+r,n^d(q-1)+s} = \tilde{K}^{d+1}_{n^d(l-1)+r,n^d(q-1)+s} \prod_{k=1}^{d} \tilde{K}^k_{n^d(l-1)+r,n^d(q-1)+s} \\
&= \tilde{K}^1_{l,q} \prod_{k=1}^{d} \tilde{K}_{r,s} = {}^{(1)}K_{l,q}\, {}^{(d)}K_{r,s}
\end{aligned}
$$

and hence ${}_{l,q}B = {}^{(1)}K_{l,q} \cdot {}^{(d)}K$ such that ${}^{(d+1)}K = {}^{(1)}K \otimes {}^{(d)}K$. □

Corollary 2.6. *The system matrix ${}^{(d)}K$ is the $d-$times Kronecker product of the $n \times n$ matrix ${}^{(1)}K$*

$$
{}^{(d)}K = {}^{(1)}K \otimes {}^{(1)}K \otimes \ldots \otimes {}^{(1)}K\,. \tag{25}
$$

To solve system (13) we use now (25) together with (34) and obtain the inverse ${}^{(d)}K^{-1}$ of the system matrix ${}^{(d)}K$.

Corollary 2.7.

$$
{}^{(d)}K^{-1} = {}^{(1)}K^{-1} \otimes {}^{(1)}K^{-1} \otimes \ldots \otimes {}^{(1)}K^{-1} \tag{26}
$$

with d Kronecker factors.

Accordingly, we can compute $c = K^{-1}C$ with the factorization (37) for solving the linear system (13). Therefore we only need to assemble the one-dimensional system matrix ${}^{(1)}K$ and have to compute its inverse. For the sake of convenience we set $A = {}^{(1)}K^{-1}$ and discuss an effective algorithm to compute the matrix vector product

$$
y = (I \otimes I \otimes \cdots \otimes I \otimes A \otimes I \otimes I \otimes \cdots \otimes I)x \tag{27}
$$

below.

2.3 Effective computation of the matrix vector multiplication

To compute a matrix vector product of equation (27) without any overhead at first we use for a unity matrix I_n of size $n \times n$

$$I_n \otimes I_n = I_{n^2}$$

and collect the p unity matrices left and q right of A to get

$$y = (I_{n^p} \otimes A \otimes I_{n^q})x$$

The I_{n^q} from the right creates a big matrix with scaled unity matrices as diagonal blocks

$$y = \left(I_{n^p} \otimes \begin{bmatrix} A_{11}I_{n^q} & A_{12}I_{n^q} & \cdots & A_{1n}I_{n^q} \\ A_{21}I_{n^q} & & & \\ \vdots & & & \vdots \\ A_{n1}I_{n^q} & A_{n2}I_{n^q} & \cdots & A_{nn}I_{n^q} \end{bmatrix} \right) x$$

The unity matrix I_{n^p} from the left creates a big block diagonal matrix with this blocks, so

$$y = \begin{bmatrix} A_{11}I_{n^q} & A_{12}I_{n^q} & \cdots & A_{1n}I_{n^q} & & & & \\ A_{21}I_{n^q} & & & & & & & \\ \vdots & & & \vdots & & & & \\ A_{n1}I_{n^q} & A_{n2}I_{n^q} & \cdots & A_{nn}I_{n^q} & & & & \\ & & & & \ddots & & & \\ & & & & & A_{11}I_{n^q} & A_{12}I_{n^q} & \cdots & A_{1n}I_{n^q} \\ & & & & & A_{21}I_{n^q} & & & \\ & & & & & \vdots & & & \vdots \\ & & & & & A_{n1}I_{n^q} & A_{n2}I_{n^q} & \cdots & A_{nn}I_{n^q} \end{bmatrix} x$$

So the matrix vector product can be computed block by block as

$$y_b = \begin{bmatrix} A_{11}I_{n^q} & A_{12}I_{n^q} & \cdots & A_{1n}I_{n^q} \\ A_{21}I_{n^q} & & & \\ \vdots & & & \vdots \\ A_{n1}I_{n^q} & A_{n2}I_{n^q} & \cdots & A_{nn}I_{n^q} \end{bmatrix} x_b$$

and the multiplication of every block is equal to

$$\begin{bmatrix} A_{11}I_{n^q} & A_{12}I_{n^q} & \cdots & A_{1n}I_{n^q} \\ A_{21}I_{n^q} & & & \\ \vdots & & & \vdots \\ A_{n1}I_{n^q} & A_{n2}I_{n^q} & \cdots & A_{nn}I_{n^q} \end{bmatrix} x_b = y_b = \begin{bmatrix} \sum_i^n A_{1i}x_b i \\ \sum_i^n A_{2i}x_b i \\ \vdots \\ \sum_i^n A_{ni}x_b i \end{bmatrix}$$

The implementation of the complete multiplication is given in the appendix at page 28, by calling

```
y = kronecker_multiplication(p,A,q,x);
```

it can be done without any overhead.

2.4 Assembling the right hand side C

The most effort needs the computation of the components

$$C_i = \int_\Omega \mathrm{C}(u)\psi_i(u)du \tag{28}$$

because for an arbitrary function $C(u)$ this only can be done numerically. With (14) and (18) equation (28) is equivalent to

$$C_i = \sum_{k=i}^{N} \int_{e_k} \mathrm{C}(u)\psi_i(u)du. \tag{29}$$

To compute the full vector C we compute for every vector component this sum of all the values of the element integrals, which is done numerically with a Gauss formula. This is the most expensive part of the computation, but it perfectly scales up, if it is done in parallel. In the implementation we use a master-slave program written in C++ with OpenMPI.

A special situation appears if we solve the statistical estimation problem for a given sample and use the empirical copula $\hat{\mathrm{C}}$ instead of C and compute a solution of the integral equation (9), since (7) as well as (17) can be decomposed into one dimensional factors and hence

$$C_i = \int_\Omega \hat{\mathrm{C}}(u)\psi_i(u)du = \frac{1}{T}\sum_{j=1}^{T}\int_{[0,1]^d}\prod_{k=1}^{d}\mathbb{1}_{\hat{U}_{kj}\le u_k}\xi(u_k, b_k^i)du$$

$$= \frac{1}{T}\sum_{j=1}^{T}\prod_{k=1}^{d}\left(\int_0^1 \mathbb{1}_{\hat{U}_{kj}\le u_k}\xi(u_k, b_k^i)du_k\right) = \frac{1}{T}\sum_{j=1}^{T}\prod_{k=1}^{d} I_{kj}^i \quad \text{with}$$

$$I_{kj}^i = \int_0^1 \mathbb{1}_{\hat{U}_{kj}\le s}\xi(u_k, b_k^i)ds = \begin{cases} h(1-b_k^i) - \frac{1}{2}h^2\,, & \hat{U}_{kj} < b_k^i \\ h(1-b_k^i) - \frac{1}{2}h^2 - \frac{1}{2}(\hat{U}_{kj} - b_k^i)^2\,, & b_k^i \le \hat{U}_{kj} \le b_k^i + h \\ h(1-\hat{U}_{kj})\,, & \hat{U}_{kj} > b_k^i + h\,. \end{cases}$$

Hence the computational effort improves from $\mathcal{O}\left(N3^dT + \frac{N^2+N}{2}3^d\right)$ using a three point Gauss formula in each dimension to $\mathcal{O}(NTd)$.

2.5 Numerical example

2.5.1 Computing times

In this numerical example we use the independent copula

$$\mathrm{C}(u) = \prod_{i=1}^{d} u_i$$

which has the exact solution $\mathrm{c}(u) = 1$ in order to illustrate the approximation quality and computing times. For the independent copula the integrated square error (see [QQX09])

$$ISE(\mathrm{c}, \mathrm{c}_h) = \int_\Omega \left(\mathrm{c}(u) - \mathrm{c}_h(u)\right)^2 du = \sum_{j=1}^{N}\int_{e_j}\left(\mathrm{c}(u) - c_j\right)^2 du \qquad (31)$$

can easily be computed

$$ISE(\mathrm{c}, \mathrm{c}_h) = \sum_{j=1}^{N}\int_{e_j}(1-c_j)^2\,du = \sum_{j=1}^{N}(1-c_j)^2\,\frac{1}{N} = \frac{1}{N}e^2 \qquad (32)$$

using the point wise error norm

$$e := \|c - [1, 1, \cdots, 1]^T\|_{l^2}\,.$$

In the table we give the the $1d$-discretization steps n, dimension d, system size $N = n^d$ computing times t_{sys} for creating the system matrix ${}^{(1)}K$, t_{rhs} for assembling the right hand side, t_{solve} for solving the system, s_{rhs} as the number of computing slaves, and the integrated square error.

For the computation of the right hand side the parallel OpenMPI implementation is used with s_{rhs} computing slaves. All other computations are done with Matlab.

d	n	N	$t_{sys}[s]$	s_{rhs}	$t_{rhs}[s]$	$t_{solve}[s]$	e	$ISE(\mathrm{c}, \mathrm{c}_h)$
2	30	900	0.7	1	0.2	0.06	1.35e-08	2.0250e-19
2	60	3600	2.7	1	2.2	0.44	4.64e-07	5.9804e-17
2	100	10000	7.6	3	6.7	1.95	4.68e-06	2.1902e-15
3	30	27000	0.7	10	60.7	1.68	1.10e-04	4.4815e-13
3	60	216000	3.4	30	1440	25.7	0.0234	2.5350e-09
3	100	1000000	7.8	30	32163	192	1.02	1.0404e-06
4	30	810000	0.7	30	72989	51	1.96	4.7427e-06

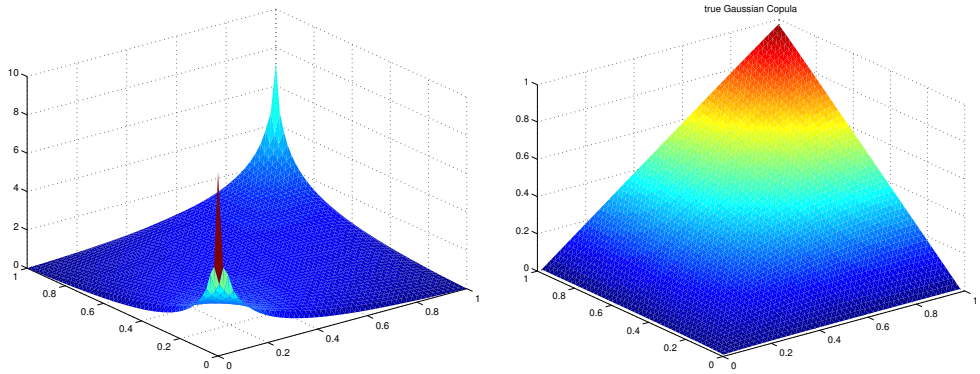
2.5.2 Reconstruction for typical copula families with exact right hand side

To show that the proposed method works quite well we present the reconstructed copula densities for exact right hand side based on the exact copula C. Figures 2, 3, 4, 5 and 6 show the reconstructed copula densities for typical copula families. For more details about copula families see [Nel06]. The choice of the copula parameters is based on the choices in [QY12] and [QQX09].

2.5.3 Noisy data: Illustration of Ill-posedness

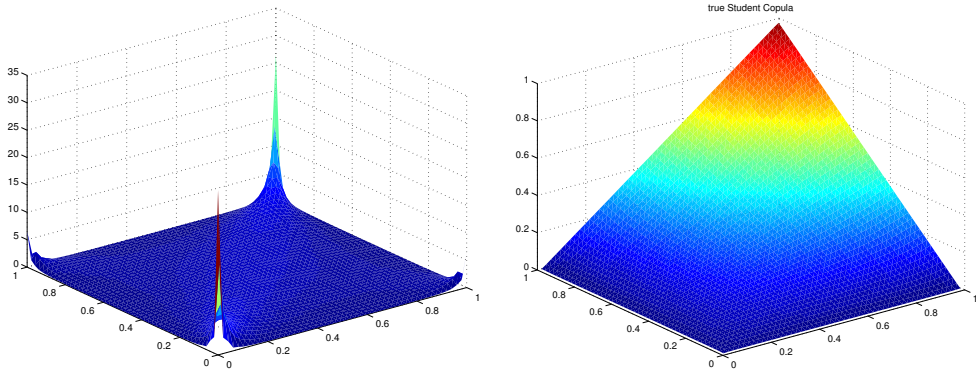
Note that in real problems the copula C is not known. For this case we have simulated T samples for each copula and present the non-parametric reconstructed densities using the Petrov-Galerkin projection. Figures 7, 8, 9, 10 and 11 illustrate the expected ill-posedness appearing for decreasing sample size T.

To overcome the ill-posedness an appropriate regularization is required. Figures 12, 13, 14, 15 and 16 show the reconstructed copula densities for $T = 1\,000$ and $T = 10\,000$ samples using the well-known Tikhonov regularization. The choice of the regularization parameter $\alpha = 10^{-8}$ is very naive



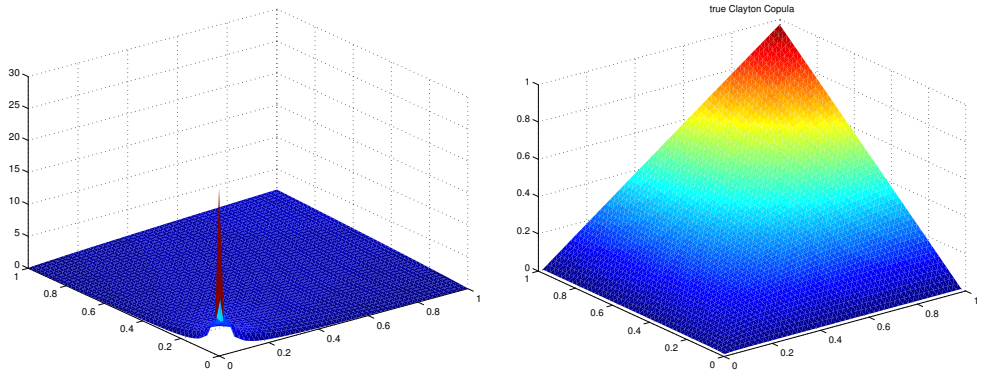
(a) reconstructed density c (b) copula C

Figure 2: Gauss copula, $\rho = 0.5$, $n = 50$



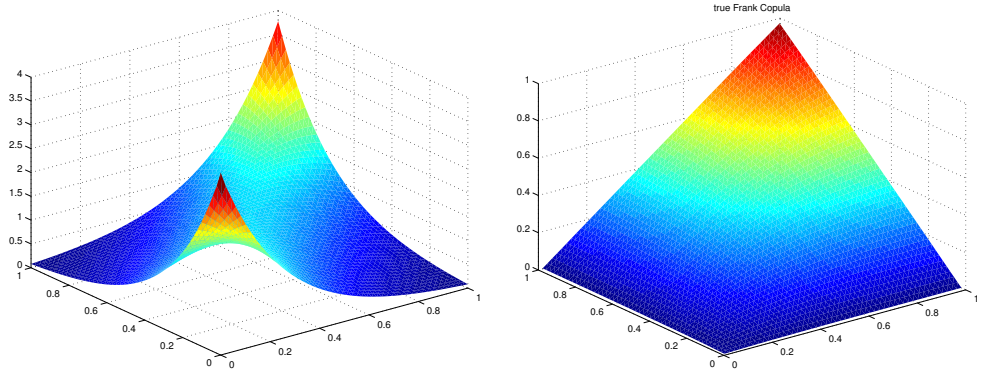
(a) reconstructed density c (b) copula C

Figure 3: Student copula, $\rho = 0.5$, $\nu = 1$, $n = 50$



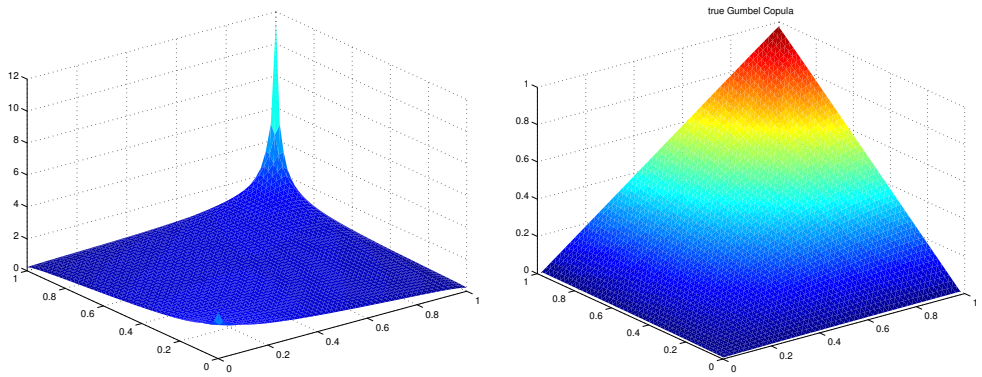
(a) reconstructed density c (b) copula C

Figure 4: Clayton copula, $\theta = 0.8$, $n = 50$



(a) reconstructed density c (b) copula C

Figure 5: Frank copula, $\theta = 4$, $n = 50$



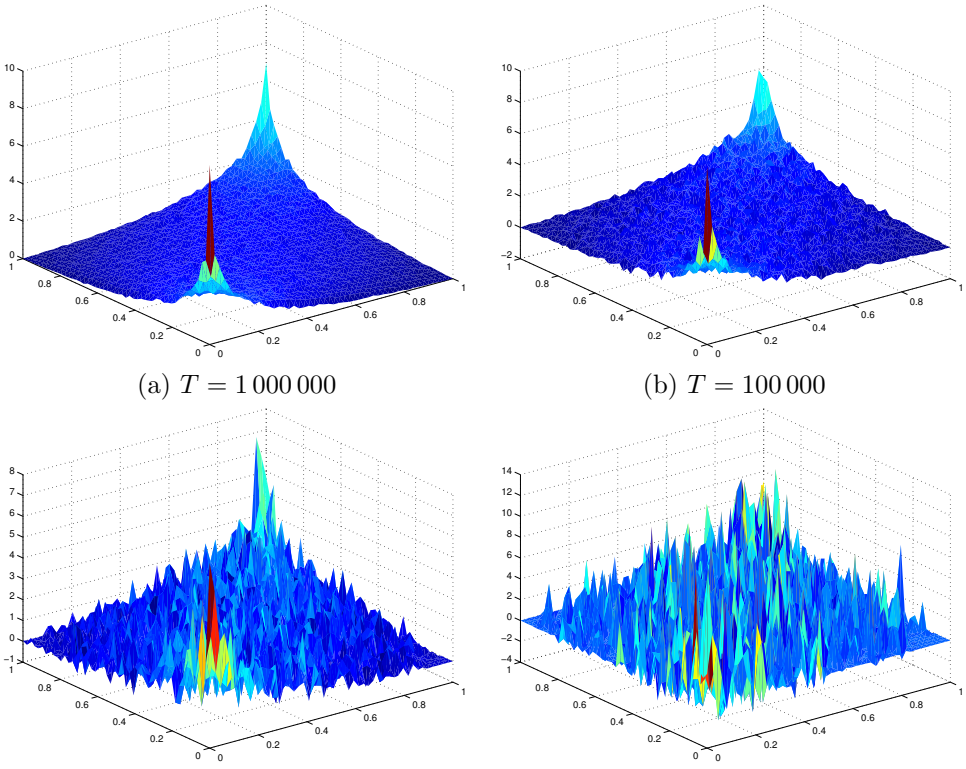
(a) reconstructed density c (b) copula C

Figure 6: Gumbel copula, $\theta = 1.25$, $n = 50$

and arbitrary and serves only as demonstration how the instability can be handled. It is further work to discuss a parameter choice rule for Tikhonov regularization as well as other regularization methods, which can use the special structure (25) of the system matrix K to avoid the complete assembling of K. In particular all regularization methods using the singular value or eigenvalue decomposition of K can be easy handled because the eigenvalue decomposition of the one dimensional matrix ${}^{(1)}K = V\Lambda V^T$ leads to the eigenvalue decomposition of the system matrix

$$K = (V \otimes \cdots \otimes V)\,(\Lambda \otimes \cdots \otimes \Lambda)\left(V^T \otimes \cdots \otimes V^T\right). \tag{33}$$

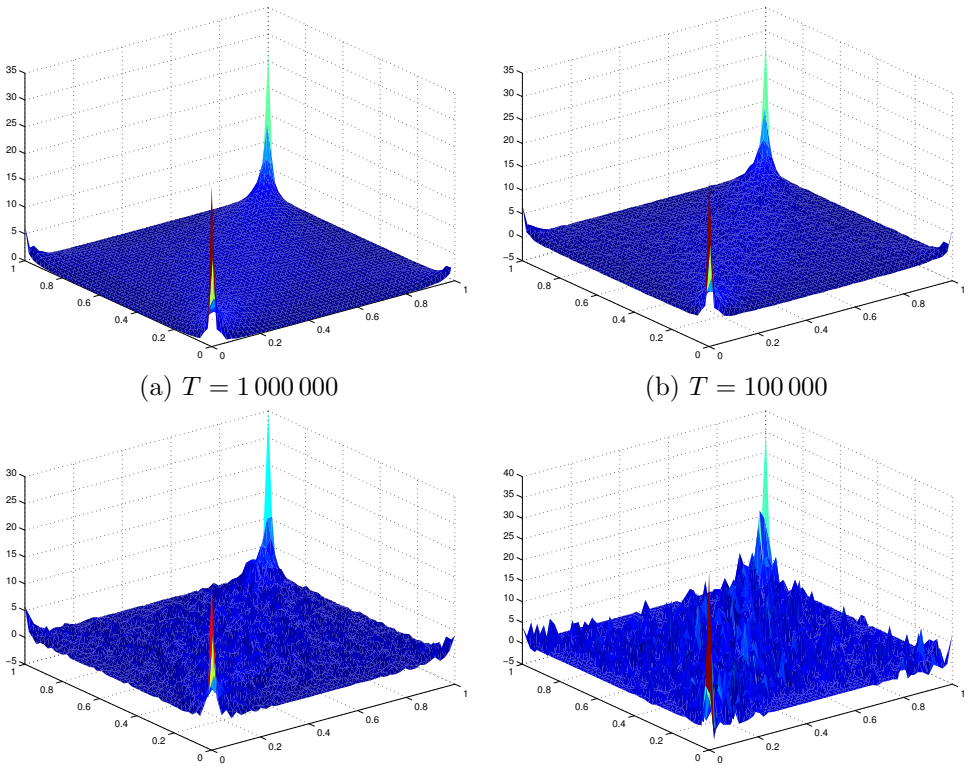
A typical property of Tikhonov regularization is, that true peaks in the density will be smoothed. Hence the reconstruction quality should be improved, if other regularization methods are used.



(a) $T = 1\,000\,000$ (b) $T = 100\,000$

(c) $T = 10\,000$ (d) $T = 1\,000$

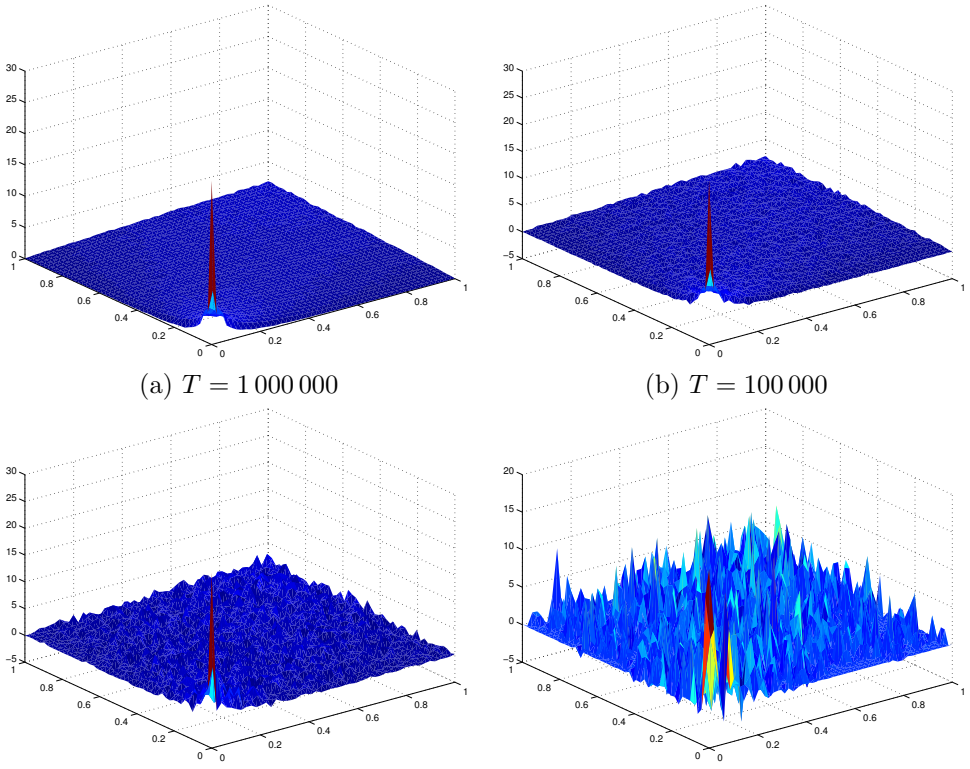
Figure 7: Gauss copula, $\rho = 0.5$, $n = 50$



(a) $T = 1\,000\,000$ (b) $T = 100\,000$

(c) $T = 10\,000$ (d) $T = 1\,000$

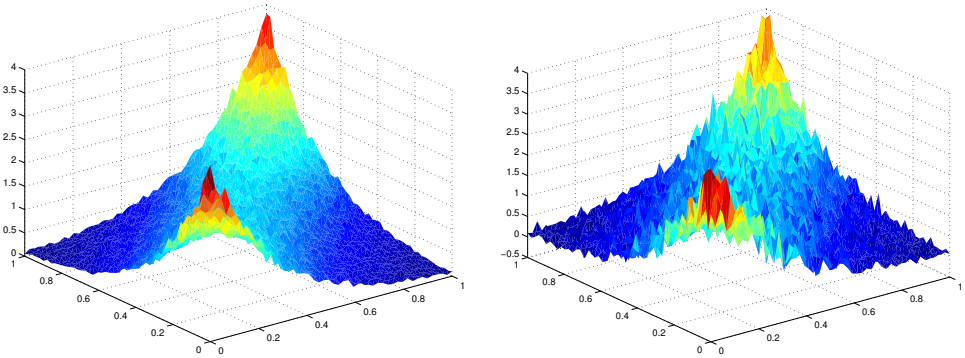
Figure 8: Student copula, $\rho = 0.5, \nu = 1$, $n = 50$



(a) $T = 1\,000\,000$ (b) $T = 100\,000$

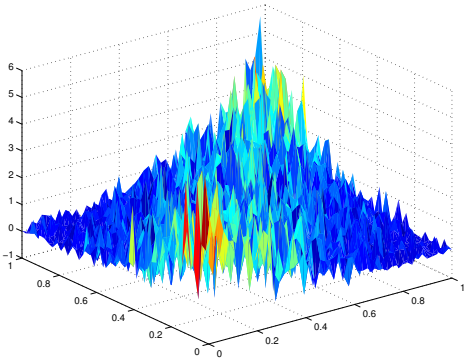
(c) $T = 10\,000$ (d) $T = 1\,000$

Figure 9: Clayton copula, $\theta = 0.8$, $n = 50$

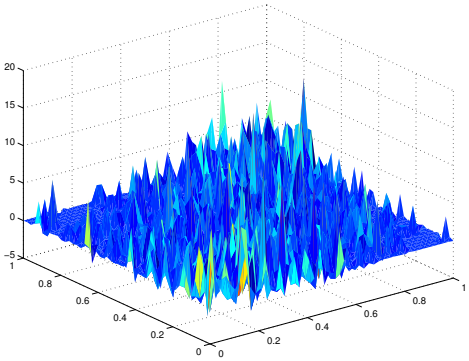


(a) $T = 1\,000\,000$

(b) $T = 100\,000$

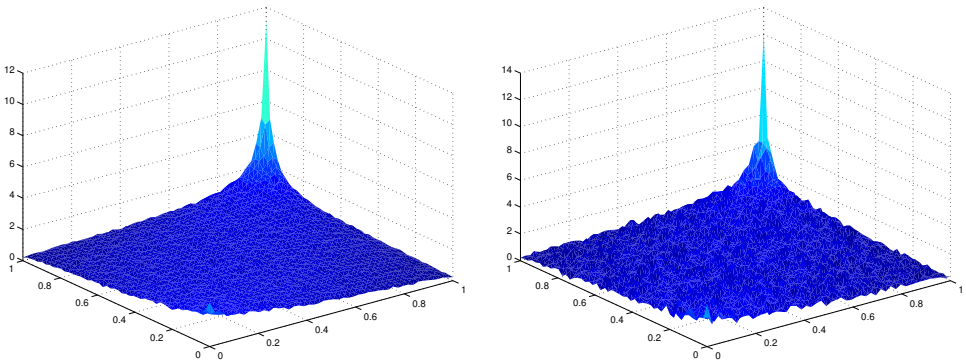


(c) $T = 10\,000$



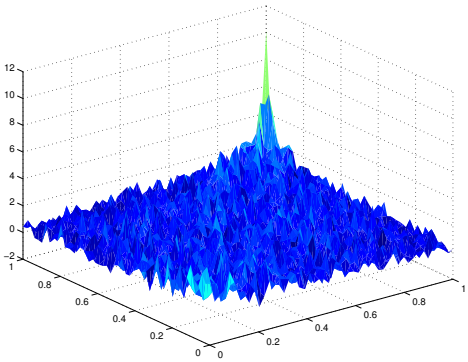
(d) $T = 1\,000$

Figure 10: Frank copula, $\theta = 4$, $n = 50$

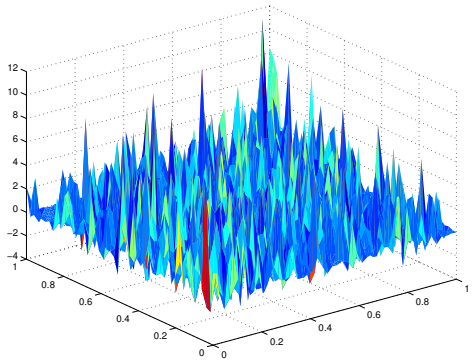


(a) $T = 1\,000\,000$

(b) $T = 100\,000$

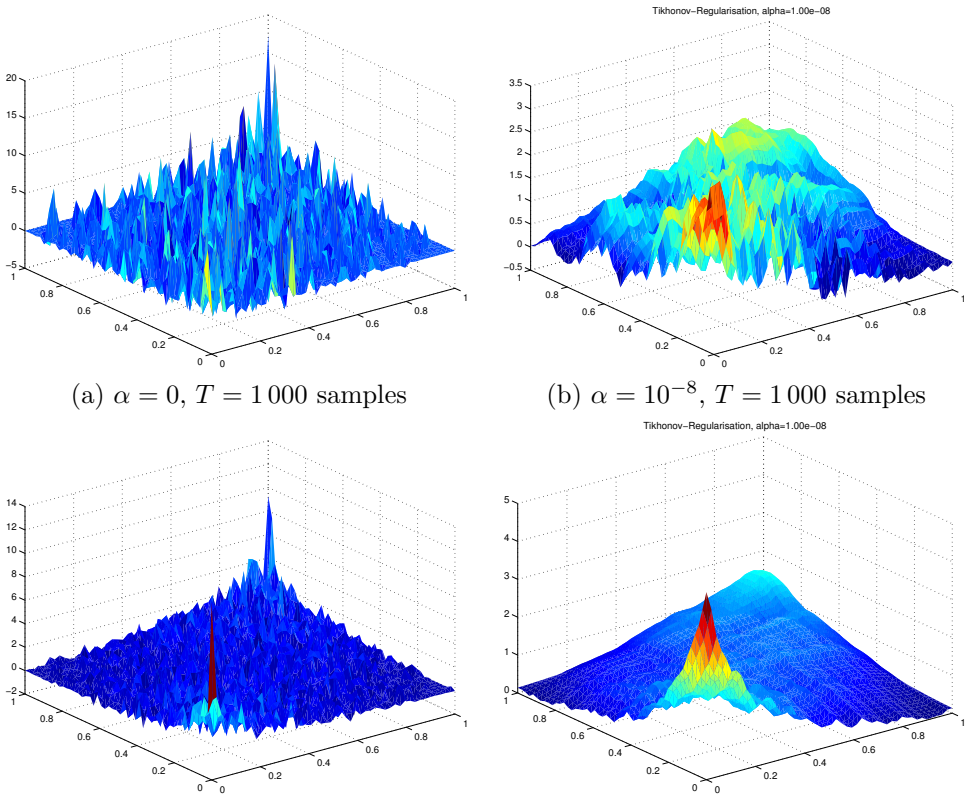


(c) $T = 10\,000$



(d) $T = 1\,000$

Figure 11: Gumbel copula, $\theta = 1.25$,$\theta = 4$, $n = 50$



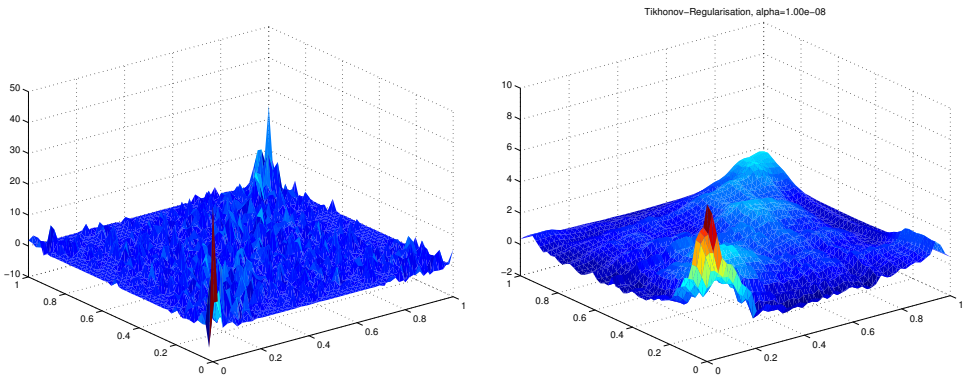
(a) $\alpha = 0$, $T = 1\,000$ samples

(b) $\alpha = 10^{-8}$, $T = 1\,000$ samples

(c) $\alpha = 0$, $T = 10\,000$ samples

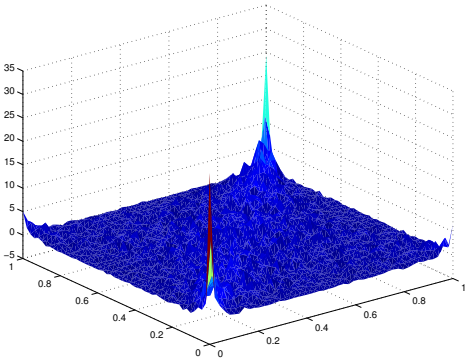
(d) $\alpha = 10^{-8}$, $T = 10\,000$ samples

Figure 12: Regularized Gauss copula, $\rho = 0.5$, $n = 50$

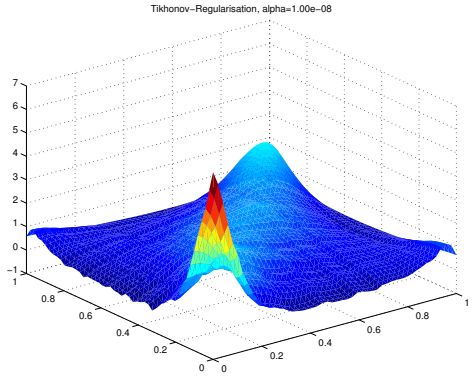


(a) $\alpha = 0$, $T = 1\,000$ samples

(b) $\alpha = 10^{-8}$, $T = 1\,000$ samples

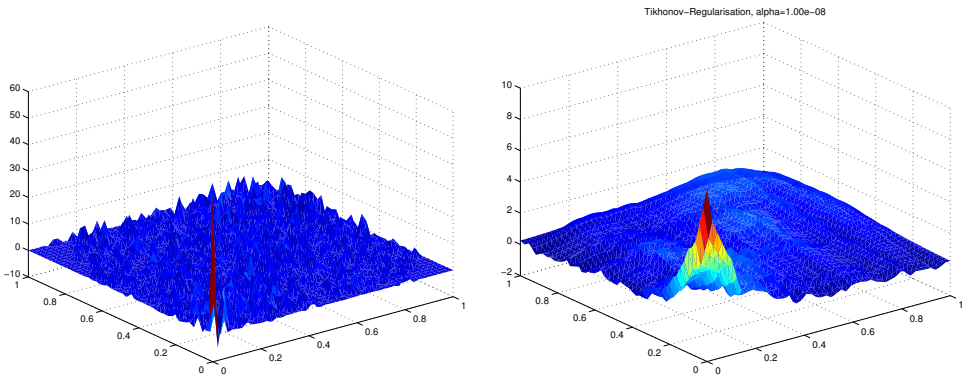


(c) $\alpha = 0$, $T = 10\,000$ samples



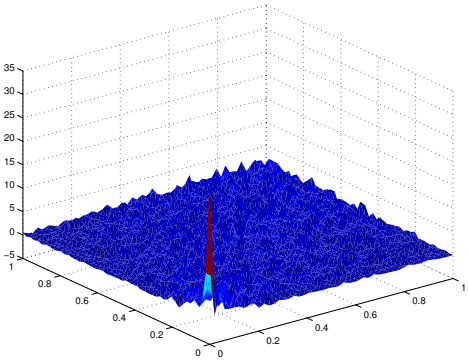
(d) $\alpha = 10^{-8}$, $T = 10\,000$ samples

Figure 13: Regularized Student copula, $\rho = 0.5, \nu = 1$, $n = 50$

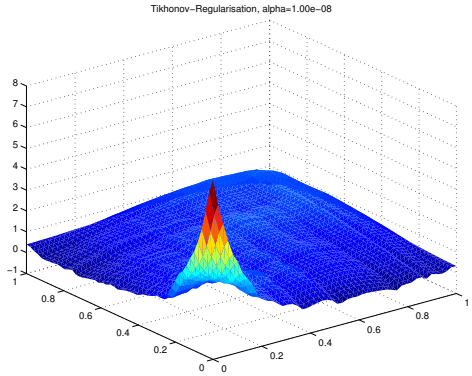


(a) $\alpha = 0$, $T = 1\,000$ samples

(b) $\alpha = 10^{-8}$, $T = 1\,000$ samples

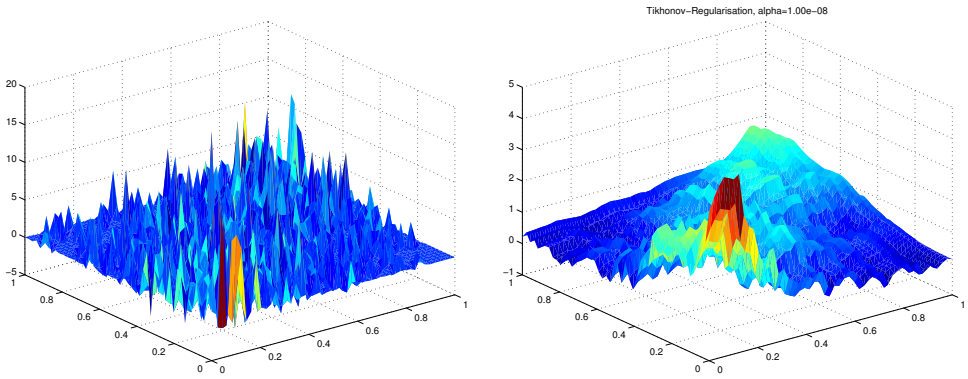


(c) $\alpha = 0$, $T = 10\,000$ samples



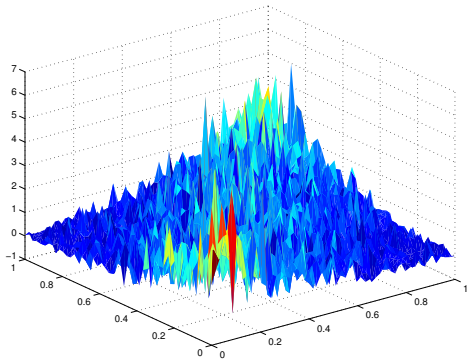
(d) $\alpha = 10^{-8}$, $T = 10\,000$ samples

Figure 14: Regularized Clayton copula, $\theta = 0.8$, $n = 50$

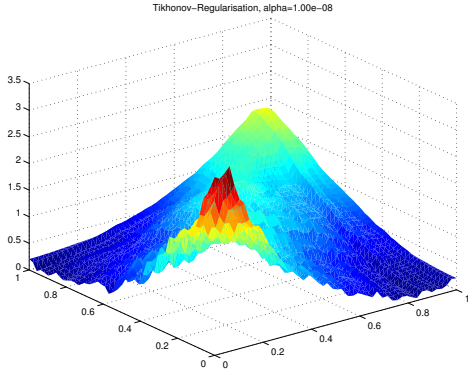


(a) $\alpha = 0$, $T = 1\,000$ samples

(b) $\alpha = 10^{-8}$, $T = 1\,000$ samples

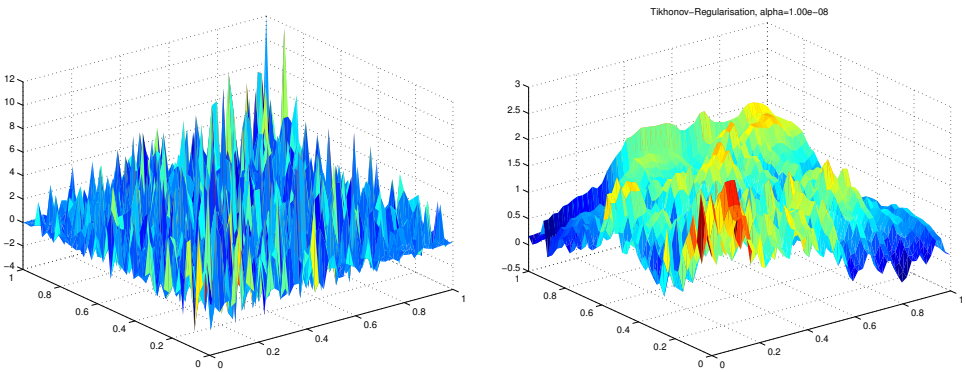


(c) $\alpha = 0$, $T = 1\,000$ samples



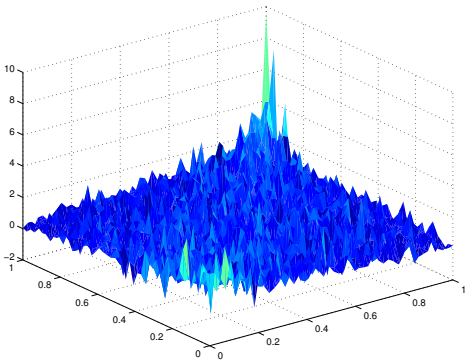
(d) $\alpha = 10^{-8}$, $T = 1\,000$ samples

Figure 15: Regularized Frank copula, $\theta = 4$, $n = 50$

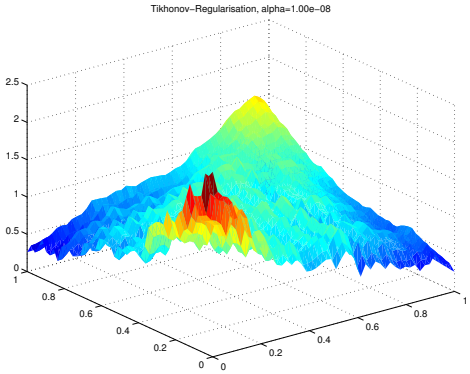


(a) $\alpha = 0$, $T = 1\,000$ samples

(b) $\alpha = 10^{-8}$, $T = 1\,000$ samples



(c) $\alpha = 0$, $T = 10\,000$ samples



(d) $\alpha = 10^{-8}$, $T = 10\,000$ samples

Figure 16: Regularized Gumbel copula, $\theta = 1.25$,$\theta = 4$, $n = 50$

A The Kronecker product

The Kronecker product of matrices A and B is the block matrix C, given by

$$C = A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B & \cdots & A_{1n}B \\ \vdots & & & \vdots \\ A_{m1}B & A_{m2}B & \cdots & A_{mn}B \end{bmatrix}$$

The Kronecker product does not commute, for a detailed description of all properties see for example [Loa00], here we use the following properties.

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \tag{34}$$
$$AB \otimes CD = (A \otimes C)(B \otimes D) \tag{35}$$

Especially with (35) by swapping B and D

$$AD \otimes CB = (A \otimes C)(D \otimes B)$$

and setting $D = I$, $C = I$ it comes up

$$A \otimes B = (A \otimes I)(I \otimes B) \tag{36}$$

The right hand side in (36) commutes, see

$$(I \otimes B)(A \otimes I) = IA \otimes BI = A \otimes B$$

and this can be repeated for more Kronecker factors than only 2, such that the following factorization of the Kronecker product holds (see [FPS98])

$$A_1 \otimes A_2 \otimes \cdots \otimes A_n = \prod_{i=1}^{n} I \otimes I \otimes \cdots \otimes I \otimes A_i \otimes I \otimes I \otimes \cdots \otimes I \tag{37}$$

with (i-1) kronecker factors I left of A_i and $(n-i)$ right of A_i.

B Effective Kronecker multiplication

The matrix vector multiplication

$$y = (I \otimes I \otimes \cdots \otimes I \otimes A \otimes I \otimes I \otimes \cdots \otimes I)x$$

with p unity matrices left and q right of A can computed without any overhead by calling `y = kronecker_multiplication(p,A,q,x);` with the following functions in matlab notation.

```
function y=kronecker_multiplication(p,A,q,x)
%
%  y= (I # I # ... # I # A # I # ... # I) x
%
%  where # is the Kronecker operator and
%  we have p Is left and q Is right from A
%
%  R. Unger 03/2013
%

[m,n]=size(A);
if (m ~= n)
 error('kronecker_multiplication: wrong dimension of A (not n x n)');
end;

N=length(x);

if (N ~= n^(p+q+1) )
 error('kronecker_multiplication: wrong dimension of x');
end;

% cut x in n^p blocks of size n^(q+1)
% and process every block separately
%
y=zeros(N,1);
for i=1:n^p
  xb = x( (i-1)*n^(q+1) +1  : i*n^(q+1));
  yb = kronecker_multiplication_block(A,q,xb);
  y( (i-1)*n^(q+1) +1  : i*n^(q+1)) = yb;
end

function y = kronecker_multiplication_block(A,q,x);
%
%  Compute y = ( A # I # ... # I) x
%
%  where # is the Kronecker operator and
%  we have q Is right from A
%
```

```
%  R. Unger 03/2013
%

[m,n]=size(A);
if (m ~= n) error('wrong dimension of A (not n x n)') ; end;

N=length(x);


if (N ~= n^(q+1) ) error('wrong dimension of x') ; end;

% we have n blocks of size n^q
y=zeros(n^(q+1),1 );
for i=1:n
  yb=zeros(n^q,1);
  for j=1:n
    yb=yb+A(i,j) * x( (j-1)*n^(q) +1  : j*n^(q));
  end
  y( (i-1)*n^(q) +1  : i*n^(q)) = yb;
end
```

With this multiplications, applied to the right hand side vector the solution vector c can be computed for high values of N in short times.

References

[Deh80] Paul Deheuvels. Non parametric tests of independence. In Jean-Pierre Raoult, editor, *Statistique non Paramétrique Asymptotique*, volume 821 of *Lecture Notes in Mathematics*, pages 95–107. Springer Berlin Heidelberg, 1980.

[FPS98] P. Fernandes, B. Plateau, and W. J. Stewart. Efficient descriptor-vector multiplication in stochastic automata networks. *Journal of the ACM (JACM)*, 45(3):381–414, May 1998.

[Loa00] Charles F.Van Loan. The ubiquitous kronecker product. *Journal of Computational and Applied Mathematics*, 123(12):85 – 100, 2000. Numerical Analysis 2000. Vol. III: Linear Algebra.

[MFE10] A.J. McNeil, R. Frey, and P. Embrechts. *Quantitative Risk Management: Concepts, Techniques, and Tools*. Princeton Series in Finance. Princeton University Press, 2010.

[MS12] J.F. Mai and M. Scherer. *Simulating Copulas: Stochastic Models, Sampling Algorithms, and Applications*. Series in Quantitative Finance. Imperial College Press, 2012.

[Nel06] Roger B. Nelsen. *An Introduction to Copulas (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[QQX09] Leming Qu, Yi Qian, and Hui Xie. Copula density estimation by total variation penalized likelihood. *Communications in Statistics - Simulation and Computation*, 38(9):1891–1908, 2009.

[QY12] Leming Qu and Wotao Yin. Copula density estimation by total variation penalized likelihood with linear equality constraints. *Computational Statistics & Data Analysis*, 56(2):384 – 398, 2012.