Efficient Gaussian Copula Density Computation for Large-Scale Spatial Data: A Matérn-like GMRF Approach with Circulant and Folded Circulant Approximations

Brynjólfur Gauti Guðrúnar Jónsson*

University of Iceland,

E-mail: brynjolfur@hi.is

Source: Article Notebook

Introduction

Problem Formulation

Consider a spatial field on a regular $n_1 \times n_2$ grid. Our objective is to compute the Gaussian copula density efficiently for this field. This computation involves:

1. Specifying an $n_1 n_2 \times n_1 n_2$ precision matrix $\mathbf{Q}$ that represents the spatial dependence structure.
2. Ensuring the implied covariance matrix $= \mathbf{Q}^{-1}$ has unit diagonal elements.
3. Computing the log determinant, $\log|\mathbf{Q}|$, and the quadratic form $z^T\mathbf{Q}z$ where $z_i = \Phi^{-1}(f_i(y_i))$

Review

Gaussian Markov Random Fields (GMRFs) and copulas are two powerful statistical tools, each offering unique strengths in modeling complex data structures. GMRFs excel in capturing spatial and temporal dependencies, particularly in fields such as environmental science, epidemiology, and image analysis. Their ability to represent local dependencies through sparse precision matrices makes them computationally attractive for high-dimensional problems. Copulas, on the other hand, provide a flexible framework for modeling multivariate dependencies, allowing separate specification of marginal distributions and their joint behavior.

The Gaussian copula, in particular, has gained popularity due to its interpretability and connection to the multivariate normal distribution. However, combining GMRFs with copulas has historically been computationally challenging, limiting their joint application to smaller datasets or simpler models.

Let $\mathbf{X} = (X_1, X_2, \dots, X_n)$ be a multivariate random vector with marginal distribution functions F_i for $i = 1, 2, \dots, n$. The joint distribution function of $\mathbf{X}$ can be written as:

$$F_{\mathbf{X}}(\mathbf{x}) = C(F_1(x_1), F_2(x_2), \dots, F_n(x_n)),$$

where C is the Gaussian copula defined by the GMRF precision matrix $\mathbf{Q}$. The Gaussian copula C is given by:

$$C(u_1, u_2, \dots, u_n) = \Phi_{\mathbf{Q}}(\Phi^{-1}(u_1), \Phi^{-1}(u_2), \dots, \Phi^{-1}(u_n)),$$

where $\Phi_{\mathbf{Q}}$ is the joint cumulative distribution function of a multivariate normal distribution with mean vector $\mathbf{0}$ and precision matrix $\mathbf{Q}$, and Φ^{-1} is the inverse of the standard normal cumulative distribution function.

A critical requirement for the precision matrix $\mathbf{Q}$ governing the GMRF copula C is that $= \mathbf{Q}^{-1}$ should have a unit diagonal, i.e. the marginal variance is equal to one everywhere.. This ensures it operates on the same scale as the transformed data, $\Phi^{-1}(u_i)$. However, this can be challenging as GMRFs are typically defined in terms of precision matrices that often imply non-unit marginal variances.

This paper presents a novel algorithm that bridges the gap between GMRFs and copulas, allowing for fast and efficient computation of Gaussian copula densities using GMRF precision structures. Our method focuses on creating a Matérn-like precision matrix $\mathbf{Q}$ with unit marginal variance and efficiently computing the multivariate Gaussian copula density of $\mathbf{Z} = \Phi^{-1}(\mathbf{u})$, where $u_i \sim \text{Uniform}(0, 1)$, $i = 1, \ldots, n$.

The key innovation lies in leveraging the special structure of the precision matrix:

$$\mathbf{Q} = \mathbf{Q}_{\rho_1} \otimes \mathbf{I}_{\mathbf{n_2}} + \mathbf{I}_{\mathbf{n_1}} \otimes \mathbf{Q}_{\rho_2},$$

where $\mathbf{Q}_\rho$ is the precision matrix of a standardized one-dimensional AR(1) process with correlation ρ and $\otimes$ denotes the Kronecker product. By employing efficient eigendecomposition techniques, our method avoids explicit formation and inversion of the large precision matrix $\mathbf{Q}$, making it particularly suitable for high-dimensional spatial data. In addition to the exact method, we show how the precision matrix can be approximated by a folded circulant matrix wich gives a large speed-up while preserving suitable boundary conditions.

Methods

Gaussian Copula Density Computation

The Gaussian copula density for a random vector $\mathbf{U} = (U_1, ..., U_n)$ with $U_i \sim \text{Uniform}(0, 1)$ is given by:

$$c(\mathbf{u}) = |\mathbf{Q}|^{1/2} \exp\left(-\frac{1}{2}\mathbf{z}^T(\mathbf{Q}-\mathbf{I})\mathbf{z}\right)$$

where $\mathbf{z} = (z_1, ..., z_n)$ with $z_i = \Phi^{-1}(u_i)$, $\mathbf{Q}$ is the precision matrix, and $\mathbf{I}$ is the identity matrix.

The log-density can be expressed as:

$$\log c(\mathbf{u}) = \frac{1}{2}\log|\mathbf{Q}| - \frac{1}{2}\mathbf{z}^T\mathbf{Q}\mathbf{z} + \frac{1}{2}\mathbf{z}^T\mathbf{z}$$

Our goal is to efficiently compute this log-density for large spatial fields.

Precision Matrix Structure

We define the precision matrix $\mathbf{Q}$ as:

$$\mathbf{Q} = (\mathbf{Q}_{\rho_1} \otimes \mathbf{I}_{\mathbf{n_2}} + \mathbf{I}_{\mathbf{n_1}} \otimes \mathbf{Q}_{\rho_2})^{(\nu+1)}, \quad \nu \in \{0, 1, 2\}$$

where $\mathbf{Q}_\rho$ is the precision matrix of a one-dimensional AR(1) process with correlation ρ:

$$\mathbf{Q}_\rho = \frac{1}{1-\rho^2}\begin{bmatrix} 1 & -\rho & 0 & \cdots & 0 \\ -\rho & 1+\rho^2 & -\rho & \cdots & 0 \\ 0 & -\rho & 1+\rho^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

The matrix, $\mathbf{Q}$, is then scaled so that its inverse, $= \mathbf{Q}^{-1}$ is a correlation matrix, i.e. $_{ii} = 1$.

Computation Process

Step 1: Eigendecomposition of $\mathbf{Q}_\rho$

We first compute the eigendecomposition of both $\mathbf{Q}_\rho$\$:

$$\mathbf{Q}_\rho = \mathbf{V} \quad \mathbf{V}^T$$

where $\mathbf{V}$ is the matrix of eigenvectors and is the diagonal matrix of eigenvalues. Then, because of how Q is defined, its eigendecomposition is:

$$\mathbf{Q} = (\mathbf{V}_{\mathbf{1}} \otimes \mathbf{V}_{\mathbf{2}})(_{\mathbf{1}} \otimes \mathbf{I} + \mathbf{I} \otimes _{\mathbf{2}})^{(\nu+1)}(\mathbf{V}_{\mathbf{1}} \otimes \mathbf{V}_{\mathbf{2}})^T.$$

We don't work with the full eigendecomposition, but rather utilize the fact that the eigenvalues of $\mathbf{Q}$ are $\left\{\lambda_{\rho_1}\right\}_i + \left\{\lambda_{\rho_2}\right\}_j$ and their corresponding eigenvectors are $\left\{\mathbf{v}_{\rho_1}\right\}_i \otimes \left\{\mathbf{v}_{\rho_2}\right\}_j$ to iterate over each value and vector pair to compute the density without forming the larger matrix.

Step 2: Computation of Marginal Standard Deviations

In order to scale $\mathbf{Q}$ so that its inverse is a correlation matrix, we first calculate $\sigma_i = \sqrt{\Sigma}_{ii}$, $i = 1, \ldots, n_1 n_2$. We then use these marginal standard deviations to scale the eigenvectors and values. The inverse of Q is given by:

$$\Sigma = \mathbf{Q}^{-1} = (VAV^T)^{-1} = VA^{-1}V$$

The diagonal elements, Σ_{ii}, are given by:

$$\Sigma_{ii} = \sum_{k=1}^{n} v_{ik} \frac{1}{\lambda_k} (v^T)_{ki} = \sum_{k=1}^{n} v_{ik} \frac{1}{\lambda_k} v_{ik} = \sum_{k=1}^{n} v_{ik}^2 \frac{1}{\lambda_k}$$

This means that the i'th marginal variance, σ_i^2, is a weighted sum of the reciprocals of the eigenvalues of $\mathbf{Q}$ where the weights are the squares of the i'th value in each eigenvector. This means that we can calculate the marginal standard deviations by iterating over the eigenvalues and -vectors of Q_{ρ_1} and Q_{ρ_2} and cumulating their values according to the formula above, then taking the element-wise square roots.

Step 3: Scaling the Eigendecomposition

To scale the eigendecomposition of $\mathbf{Q}$ using the marginal standard deviations, we define a diagonal matrix $\mathbf{D}$, where $D_{ii} = \sigma_i$ and scale the precision matrix as:

$$\begin{aligned}\tilde{\mathbf{Q}} &= \mathbf{D}\mathbf{Q}^{\nu+1}\mathbf{D} \\ &= \mathbf{D}\mathbf{V}^{\ \nu+1}\mathbf{V}^T\mathbf{D} \\ &= \tilde{\mathbf{V}}\,\tilde{\mathbf{V}}^T.\end{aligned}$$

In practice, we don't scale the whole eigendecomposition. Instead, we rescale each value/vector pair individually as we iterate over the eigenvectors and values of Q_{ρ_1} and Q_{ρ_2} to create the corresponding values and vectors for the larger matrix.

Step 4: Efficient Computation of Log-Density

Using this scaled eigendecomposition, we efficiently compute:

1. Log-determinant: $\log|\tilde{\mathbf{Q}}| = \sum_{i,j}\log(\tilde{\lambda}_{ij})$, where $\tilde{\lambda}_{ij}$ is $\left(\left\{\lambda_{\rho_1}\right\}_i + \left\{\lambda_{\rho_2}\right\}_j\right)^{\nu+1}$ after rescaling with marginal standard deviations.
2. Quadratic form: $\mathbf{z}^T\tilde{\mathbf{Q}}\mathbf{z} = \sum_{i,j}(\tilde{\lambda}_{ij})y_{ij}^2$, where $y_{ij} = \left(\left\{\mathbf{v}_{\rho_1}\right\}_i \otimes \left\{\mathbf{v}_{\rho_2}\right\}_j\right)^T\mathbf{z}$.

This approach allows us to calculate the density of the spatial copula by calculating and iterating over the spectral decomposition of the smaller matrices, avoiding the formation of $\mathbf{Q}$ alltogether.

Circulant and Folded Circulant Approximations

While the eigendecomposition method provides an exact solution, it can be computationally expensive for very large spatial fields. To address this, we introduce circulant and folded circulant approximations that offer potential computational advantages.

Circulant Matrices

A circulant matrix C is a special kind of matrix where each row is a cyclic shift of the row above it. It can be fully specified by its first row or column, called the base c:

$$C = \begin{pmatrix} c_0 & c_1 & c_2 & \cdots & c_{n-1} \\ c_{n-1} & c_0 & c_1 & \cdots & c_{n-2} \\ c_{n-2} & c_{n-1} & c_0 & \cdots & c_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & c_3 & \cdots & c_0 \end{pmatrix} = (c_{j-i \mod n})$$

The base vector c completely determines the circulant matrix and plays a crucial role in efficient computations. In particular:

1. The eigenvalues of C are given by the Discrete Fourier Transform (DFT) of c:

$$\lambda = \mathrm{DFT}(c)$$

2. Matrix-vector multiplication can be performed using the FFT:

$$Cv = \mathrm{DFT}(\mathrm{DFT}(c) \odot \mathrm{IDFT}(v))$$

3. When C is non singular, then the inverse is circulant and thus determined by its base:

$$\frac{1}{n}\text{IDFT}(\text{DFT}(c)^{-1}).$$

These properties allow for much faster computations than for general matrices.

Block Circulant Matrices

For two-dimensional spatial fields, we use block circulant matrices with circulant blocks (BCCB). An $Nn \times Nn$ matrix C is block circulant if it has the form:

$$C = \begin{pmatrix} C_0 & C_1 & C_2 & \cdots & C_{N-1} \\ C_{N-1} & C_0 & C_1 & \cdots & C_{N-2} \\ C_{N-2} & C_{N-1} & C_0 & \cdots & C_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_1 & C_2 & C_3 & \cdots & C_0 \end{pmatrix} = (C_{j-i \mod N})$$

where each C_i is itself a circulant $n \times n$ matrix.

For a BCCB matrix, we define a base matrix $\mathbf{c}$, which is an $n \times N$ matrix where each column is the base vector of the corresponding circulant block. This base matrix $\mathbf{c}$ completely determines the BCCB matrix and is central to efficient computations:

1. The eigenvalues of C are given by the 2D DFT of $\mathbf{c}$.
2. Matrix-vector multiplication can be performed using the 2D FFT.
3. When C is non singular, then the inverse is also a BCCB matrix and thus determined by its base matrix.

Approximations for Q_ρ

Let Q_ρ be the precision matrix of a one-dimensional AR(1) process with correlation ρ. The exact form of Q_ρ is:

$$\mathbf{Q}_\rho = \frac{1}{1-\rho^2}\begin{bmatrix} 1 & -\rho & 0 & \cdots & 0 \\ -\rho & 1+\rho^2 & -\rho & \cdots & 0 \\ 0 & -\rho & 1+\rho^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Circulant Approximation

The circulant approximation to Q_ρ, denoted as $\mathbf{Q}_\rho^{(circ)}$, is:

$$\mathbf{Q}_\rho^{(circ)} = \frac{1}{1-\rho^2}\begin{bmatrix} 1+\rho^2 & -\rho & 0 & \cdots & 0 & -\rho \\ -\rho & 1+\rho^2 & -\rho & \cdots & 0 & 0 \\ 0 & -\rho & 1+\rho^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -\rho & 0 & 0 & \cdots & -\rho & 1+\rho^2 \end{bmatrix}$$

This approximation treats the first and last observations as neighbors, effectively wrapping the data around a circle.

Folded Circulant Approximation

The folded circulant approximation, $\mathbf{Q}_\rho^{(fold)}$, is based on a reflected version of the data. We double the data by reflecting it, giving us the data $x_1, \ldots, x_n, x_n, \ldots, x_1$. We then model this doubled data with a $2n \times 2n$ circulant matrix. If written out as an $n \times n$ matrix, it takes the form:

$$\mathbf{Q}_{\rho}^{(fold)} = \frac{1}{1-\rho^2}\begin{bmatrix} 1-\rho+\rho^2 & -\rho & 0 & \cdots & 0 & 0 \\ -\rho & 1+\rho^2 & -\rho & \cdots & 0 & 0 \\ 0 & -\rho & 1+\rho^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\rho & 1-\rho+\rho^2 \end{bmatrix}$$

This approximation modifies the first and last diagonal elements to account for the reflection of the data. As x_1 now is the first and last data point, then we avoid the circular dependence from the regular circulant approximation.

Extension to the Full Q Matrix

For a two-dimensional spatial field on an $n_1 \times n_2$ grid, we construct the full precision matrix Q using a Kronecker sum:

$$\mathbf{Q} = \left(\mathbf{Q}_{\rho_1} \otimes \mathbf{I}_{\mathbf{n_2}} + \mathbf{I}_{\mathbf{n_1}} \otimes \mathbf{Q}_{\rho_2}\right)^{(\nu+1)}, \quad \nu \in \{0, 1, 2\}$$

where $\otimes$ denotes the Kronecker product, I_n is the $n \times n$ identity matrix, and ν is a smoothness parameter.

When we approximate Q_ρ with a circulant matrix, this Kronecker sum results in a block-circulant matrix with circulant blocks (BCCB). To see this, let's consider the case where $\nu = 0$ for simplicity:

$$\mathbf{Q} = \mathbf{Q}_{\rho_1} \otimes \mathbf{I}_{\mathbf{n_2}} + \mathbf{I}_{\mathbf{n_1}} \otimes \mathbf{Q}_{\rho_2}$$

Now, let the two AR(1) matrices be approximated by a circulant matrix C with base vector $c = [c, c_1, ..., c_{n-1}]$. Then:

$$\mathbf{Q}_{\rho_1} \approx \mathbf{C}_{\mathbf{1}} = \frac{1}{1-\rho_1^2}\begin{bmatrix} 1+\rho_1^2 & -\rho_1 & 0 & \cdots & 0 & -\rho_1 \\ -\rho_1 & 1+\rho_1^2 & -\rho_1 & \cdots & 0 & 0 \\ 0 & -\rho_1 & 1+\rho_1^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -\rho_1 & 0 & 0 & \cdots & -\rho_1 & 1+\rho_1^2 \end{bmatrix},$$

and C_{ρ_2} is defined similarly. The Kronecker product $\mathbf{C}_{\rho_1} \otimes \mathbf{I}_{n_2}$ results in a block matrix where each block is a scalar multiple of I_{n_2}:

$$\mathbf{C}_{\mathbf{1}} \otimes \mathbf{I}_{\mathbf{n_2}} = \frac{1}{1-\rho_1^2}\begin{pmatrix} (1+\rho_1^2)\mathbf{I}_{\mathbf{n_2}} & -\rho_1\mathbf{I}_{\mathbf{n_2}} & \cdots & \cdots & -\rho_1\mathbf{I}_{\mathbf{n_2}} \\ -\rho_1\mathbf{I}_{\mathbf{n_2}} & (1+\rho_1^2)\mathbf{I}_{\mathbf{n_2}} & -\rho_1\mathbf{I}_{\mathbf{n_2}} & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & -\rho_1\mathbf{I}_{\mathbf{n_2}} & (1+\rho_1^2)\mathbf{I}_{\mathbf{n_2}} & -\rho_1\mathbf{I}_{\mathbf{n_2}} \\ -\rho_1\mathbf{I}_{\mathbf{n_2}} & \cdots & \cdots & -\rho_1\mathbf{I}_{\mathbf{n_2}} & (1+\rho_1^2)\mathbf{I}_{\mathbf{n_2}} \end{pmatrix}.$$

Similarly, $\mathbf{I}_{n_1} \otimes \mathbf{C}_{\rho_2}$ results in a block diagonal matrix where each block is a copy of C_{ρ_2}:

$$\mathbf{I}_{\mathbf{n_1}} \otimes \mathbf{C}_{\mathbf{2}} = \begin{pmatrix} \mathbf{C}_{\mathbf{2}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{\mathbf{2}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{C}_{\mathbf{2}} \end{pmatrix}.$$

The sum of these two matrices is a block-circulant matrix with circulant blocks:

$$\mathbf{Q} \approx \mathbf{C}_{\rho_1} \otimes \mathbf{I}_{n_2} + \mathbf{I}_{n_1} \otimes \mathbf{C}_{\rho_2} = \begin{pmatrix} \mathbf{B}_0 & \mathbf{B}_1 & \cdots & \mathbf{B}_{n_1-1} \\ \mathbf{B}_{n_1-1} & \mathbf{B}_0 & \cdots & \mathbf{B}_{n_1-2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_1 & \mathbf{B}_2 & \cdots & \mathbf{B}_0 \end{pmatrix}$$

where each $\mathbf{B_i}$ is a circulant matrix. Specifically,

$$\mathbf{B_0} = \frac{(1+\rho_1^2)}{(1-\rho_1^2)}\mathbf{I_{n_2}} + \mathbf{C}_{\rho_2}, \quad \text{and}$$
$$\mathbf{B_1} = \mathbf{B_{n_1-1}} = \frac{-\rho_1}{(1-\rho_1^2)}\mathbf{I_{n_2}}.$$

This BCCB structure allows us to use 2D FFT for efficient computations. The base matrix $\mathbf{c}$ for this BCCB structure is:

$$\mathbf{c} = \begin{bmatrix} \frac{(1+\rho_1^2)}{(1-\rho_1^2)} + \frac{(1+\rho_2^2)}{(1-\rho_2^2)} & \frac{-\rho_1}{(1-\rho_1^2)} & 0 & \cdots & \frac{-\rho_1}{(1-\rho_1^2)} \\ \frac{-\rho_2}{(1-\rho_2^2)} & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{-\rho_2}{(1-\rho_2^2)} & 0 & 0 & \cdots & 0 \end{bmatrix}$$

This base matrix c captures the structure of the precision matrix $\mathbf{Q}$ and allows for efficient computation of eigenvalues using the 2D Fast Fourier Transform (FFT), enabling rapid calculation of the log-determinant and quadratic forms needed for the Gaussian copula density.

Computation with Circulant Approximation

When using the circulant approximation, we leverage the efficient computation properties of block circulant matrices with circulant blocks (BCCB). This approach significantly reduces the computational complexity, especially for large spatial fields. Here's the step-by-step

process:

1. Construct the Base Matrix

First, we construct the base matrix $\mathbf{c}$ for our BCCB approximation of $\mathbf{Q}$. For an $n_1 \times n_2$ grid, $\mathbf{c}$ is an $n_2 \times n_1$ matrix:

$$\mathbf{c} = \begin{bmatrix} \frac{(1+\rho_1^2)}{(1-\rho_1^2)} + \frac{(1+\rho_2^2)}{(1-\rho_2^2)} & \frac{-\rho_1}{(1-\rho_1^2)} & 0 & \cdots & \frac{-\rho_1}{(1-\rho_1^2)} \\ \frac{-\rho_2}{(1-\rho_2^2)} & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{-\rho_2}{(1-\rho_2^2)} & 0 & 0 & \cdots & 0 \end{bmatrix}$$

This base matrix encapsulates the structure of our Matérn-like precision matrix.

2. Compute Initial Eigenvalues

We compute the initial eigenvalues of $\mathbf{Q}$ using the 2D Fast Fourier Transform (FFT) of $\mathbf{c}$:

$$\Lambda = \text{FFT2}(\mathbf{c})^{\nu+1}$$

where is the smoothness parameter.

3. Compute Marginal Variance and Rescale Eigenvalues

An important property of Block Circulant with Circulant Blocks (BCCB) matrices is that the inverse of a BCCB matrix is also a BCCB matrix, and the marginal variance is the first element in its first circulant block. We use this to efficiently compute the marginal variance and rescale the eigenvalues:

a. Compute the element-wise inverse of Λ: $^{\mathbf{inv}} = 1/\Lambda$

b. Compute the base of $\mathbf{Q}^{-1}$ using inverse 2D FFT: $\mathbf{c_{inv}} = \text{IFFT2}(\ ^{\mathbf{inv}})$

c. The marginal variance is given by the first element of $\mathbf{c^{inv}}$: $\sigma^2 = \mathbf{c^{inv}}_{(0,0)}$

d. Rescale the eigenvalues: $\widetilde{\Lambda} = \sigma^2 \Lambda$

This process ensures that the resulting precision matrix will have unit marginal variances, as required for the Gaussian copula.

4. Compute Log-Determinant

The log-determinant of the scaled $\widetilde{\mathbf{Q}}$ can be efficiently calculated as the sum of the logarithms of the scaled eigenvalues:

$$\log |\mathbf{Q}| = \sum_{i,j} \log(\widetilde{\Lambda}_{ij})$$

5. Compute Quadratic Form

To compute the quadratic form $\mathbf{z}^T\mathbf{Q}\mathbf{z}$, we use the following steps:

a. Compute the 2D FFT of z: $\hat{\mathbf{z}} = \text{FFT2}(\mathbf{z})$

b. Multiply element-wise with the scaled eigenvalues: $\hat{\mathbf{y}} = \widetilde{\Lambda} \odot \hat{\mathbf{z}}$

c. Compute the inverse 2D FFT: $\mathbf{y} = \text{IFFT2}(\hat{\mathbf{y}})$

d. Compute the dot product: $\mathbf{z}^T\mathbf{Q}\mathbf{z} = \mathbf{z}^T\mathbf{y}$

6. Compute the Log-Density

Finally, we can compute the log-density of the Gaussian copula:

$$\log c(\mathbf{u}) = \frac{1}{2} \log |\mathbf{Q}| - \frac{1}{2}\mathbf{z}^T\mathbf{Q}\mathbf{z} + \frac{1}{2}\mathbf{z}^T\mathbf{z}$$

where $\mathbf{z} = \Phi^{-1}(\mathbf{u})$.

Computation with Folded Circulant Approximation

The folded circulant approximation offers an alternative approach that can provide better accuracy near the edges of the spatial field. This method is based on the idea of reflecting the data along each coordinate axis, effectively doubling the size of the field. Other than that, the algorithmic implementation is the same except that the circulant approximation matrices to $\mathbf{Q}_\rho$ are now $2n \times 2n$.

First, we reflect the data along each coordinate axis. For a 2D spatial field represented by an $n \times n$ matrix, the reflected data takes the form:

$$\begin{bmatrix} x_{11} & \cdots & x_{1n_2} & x_{1n_2} & \cdots & x_{11} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{n_1 1} & \cdots & x_{n_1 n_2} & x_{n_1 n_2} & \cdots & x_{n_1 1} \\ x_{n_1 1} & \cdots & x_{n_1 n_2} & x_{n_1 n_2} & \cdots & x_{n_1 1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{11} & \cdots & x_{1n_2} & x_{1n_2} & \cdots & x_{11} \end{bmatrix}$$

This reflection creates a $2n_1 \times 2n_2$ matrix. The matrix is then stacked in lexicographic order before entering into the quadratic forms.

Results

Computational Efficiency

Table 1 presents the results of a benchmark comparing the time it takes to evaluate the gaussian copula density described above. For each grid size, we report the computation time for the exact method and the two approximations, along with the speed-up factor relative to the exact method. Each calculation was performed twenty times and the median times are shown in the table. The Cholesky method is described in the appendix.

		Eigen		Circulant		Folded	
Grid Size	Cholesky	Time	Speed-up	Time	Speed-Up	Time	Speed-Up
10x10	73.04µs	173.43µs	0.42x	31.9µs	2.29x	44.16µs	1.65x
20x20	1.43ms	267.01µs	5.36x	44µs	32.54x	151.74µs	9.44x
30x30	9.88ms	808.5µs	12.22x	110.2µs	89.63x	243.07µs	40.64x
40x40	37.88ms	2.06ms	18.35x	141.5µs	267.64x	393.79µs	96.18x
50x50	106.28ms	4.56ms	23.33x	188.5µs	563.82x	604.87µs	175.7x
60x60	253.66ms	8.79ms	28.86x	242.4µs	1046.59x	836.07µs	303.4x
70x70	538.42ms	16.81ms	32.04x	333.1µs	1616.37x	1.17ms	459.89x
80x80	1.02s	29.59ms	34.63x	393.9µs	2601.89x	1.59ms	643.46x
90x90	1.81s	48.56ms	37.27x	598.9µs	3022.23x	2.04ms	888.28x
100x100	3.09s	76.03ms	40.59x	593µs	5204.38x	2.42ms	1276.46x

Source: Article Notebook

Appendix

Cholesky Methods

Standard methods of evaluating multivariate normal densities using the Cholesky decomposition were implemented to compare with the new methods for benchmarking.

Unscaled Precision Matrix

Precision Matrix Construction

We start by constructing the precision matrix Q for a 2D Matérn field on a grid of size $d_x \times d_y$:

$$Q = Q_1 \otimes I_{d_y} + I_{d_x} \otimes Q_2$$

where $\otimes$ denotes the Kronecker product, Q_1 and Q_2 are 1D precision matrices for the x and y dimensions respectively (typically AR(1)-like structures), and I_{d_x} and I_{d_y} are identity matrices of appropriate sizes.

Density Computation

For a Matérn field with smoothness parameter ν, we need to work with $Q^{\nu+1}$. We can efficiently compute the log-determinant, $\log|Q^{\nu+1}|$, and the quadratic form, $x^T Q^{\nu+1} x$, without explicitly forming $Q^{\nu+1}$. To do this, we compute the Cholesky decomposition, $Q = LL^T$, where L is a lower triangular matrix, and make use of the following equations:

$$\log|Q^{\nu+1}| = (\nu+1)\log|Q| = 2(\nu+1)\sum_i \log(L_{ii}),$$

$$\begin{aligned} x^T Q x &= x^T L L^T x = ||L^T x||_2^2 \\ x^T Q^2 x &= x^T L L^T L L^T x = ||L L^T x||_2^2 \\ x^T Q^3 x &= x^T L L^T L L^T L L^T x = ||L^T L L^T x||_2^2. \end{aligned}$$

Algorithm

1. Construct $Q = Q_1 \otimes I_{d_y} + I_{d_x} \otimes Q_2$
2. Compute Cholesky decomposition $Q = LL^T$
3. Compute log-determinant: $\log|Q^{\nu+1}| = 2(\nu+1)\sum_i \log(L_{ii})$
4. For each observation x:
   i) Initialize $y = x$
   ii) For j from 0 to ν:
   - If j is even: $y = L^T y$

- If j is odd: $y = Ly$

iii) Compute quadratic form $q = y^T y$

5. Compute log-density: $\log p(x) = -\frac{1}{2}(d \log(2\pi) + \log |Q^{\nu+1}| + q)$

Scaled Precision Matrix

Precision Matrix Construction

We start by constructing the precision matrix Q for a 2D Matérn field on a grid of size $d_x \times d_y$:

$$Q = Q_1 \otimes I_{d_y} + I_{d_x} \otimes Q_2$$

where $\otimes$ denotes the Kronecker product, Q_1 and Q_2 are 1D precision matrices for the x and y dimensions respectively (typically AR(1)-like structures), and I_{d_x} and I_{d_y} are identity matrices of appropriate sizes. We will then have to work with the matrix $Q^{\nu+1}$.

To ensure unit marginal variances, we need to scale this precision matrix. Let D be a diagonal matrix where $D_{ii} = \sqrt{\Sigma_{ii}}$, and $\Sigma = (Q^{\nu+1})^{-1}$. The scaled precision matrix is then:

$$\tilde{Q} = DQ^{\nu+1}D$$

Efficient Computation of Scaling Matrix D

1. Compute the Cholesky decomposition of the original $Q = LL^T$
2. Compute $R = L^{-1}$, so that $S = Q^{-1} = R^T R$.
3. We then calculate the entries in D using the following steps:
   i. For $\nu = 0$, $D_{ii} = \sqrt{\Sigma_{ii}} = \sqrt{\sum_j (R_{ji})^2}$, the column-wise norm of R.
   ii. For $\nu = 1$, we use the column-wise norm of $R^T R$
   iii. For $\nu = 2$, we use the column-wise norm of $RR^T R$

Log determinant

1. First, note that $\log|\tilde{Q}| = \log|DQ^{\nu+1}D| = 2\log|D| + \log|Q^{\nu+1}|$
2. We can compute $\log|D|$ directly from the diagonal elements of D, i.e. $\log|D| = \sum_i \log(D_{ii})$
3. For $\log|Q^{\nu+1}|$, we can use the properties of the Cholesky decomposition: $\log|Q^{\nu+1}| = (\nu+1)\log|Q| = (\nu+1)\log|LL^T| = 2(\nu+1)\sum_i \log(L_{ii})$
4. Combining these, we get $\log|\tilde{Q}| = 2\sum_i \log(D_{ii}) + 2(\nu+1)\sum_i \log(L_{ii})$

Quadratic Form

1. First, note that $z^T\tilde{Q}z = z^T DQ^{\nu+1}Dz = (Dz)^T Q^{\nu+1}(Dz)$
2. Let $y = Dz$. We can compute this element-wise as $y_i = D_{ii}z_i$
3. Now we compute $y^T Q^{\nu+1} y$ as in the unscaled case.

Algorithm

Putting it all together, here's the algorithm for computing the log-density of the Gaussian copula using the scaled precision matrix:

1. Construct $Q = Q_1 \otimes I_{d_y} + I_{d_x} \otimes Q_2$
2. Compute Cholesky decomposition $Q = LL^T$
3. Compute $R = L^{-1}$ and use it to compute D as described earlier
4. Compute log-determinant: $\log|\tilde{Q}| = 2\sum_i \log(D_{ii}) + 2(\nu+1)\sum_i \log(L_{ii})$
5. For each observation $z = \Phi^{-1}(u)$:
   i) Compute $y = Dz$
   ii) Compute $y^T Q^{\nu+1} y$ as in the unscaled case.
6. Compute log-density: $\log c(u) = -\frac{1}{2}(d\log(2\pi) + \log|\tilde{Q}| + q - z^T z)$