Bridging Gaussian Markov Random Fields and Copulas: A Fast Algorithm for Efficient Gaussian Copula Density Computation with Matérn-like Precision Matrices

Brynjólfur Gauti Guðrúnar Jónsson*

University of Iceland,

E-mail: brynjolfur@hi.is

Abstract

Gaussian Markov Random Fields (GMRFs) have long been a powerful tool for modeling spatial and temporal dependencies in various fields. Similarly, copulas have proven invaluable for modeling complex dependency structures in multivariate data. However, the combination of these two approaches - using GMRFs within copula models - has historically been computationally inefficient, limiting their joint application to smaller datasets or simpler models. This work presents a novel algorithm that overcomes these limitations, allowing for fast and efficient computation of Gaussian Copula densities using GMRF precision structures. By bridging the gap between GMRFs and copulas, this method opens up new possibilities for analyzing large-scale, high-dimensional spatial and spatio-temporal data with complex dependency structures.

Introduction

Problem Formulation

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

1. Specifying a 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 density for large spatial fields in a computationally efficient manner.

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

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

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

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

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:

$$\mathbf{Q} = \mathbf{Q}_1 \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{Q}_1,$$

where $\mathbf{Q}_1$ is the precision matrix of a standardized one-dimensional AR(1) process 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 additions to the exact method, we

mention approximations to $\mathbf{Q}$ using circuland and folded circulant matrices.

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:

$$\mathbf{Q} = (\mathbf{Q}_1 \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{Q}_1)^{(\nu+1)}, \quad \nu \in \{0, 1, 2\}$$

where $\mathbf{Q}_1$ is the precision matrix of a one-dimensional AR(1) process:

$$\mathbf{Q}_1 = \frac{1}{1-\rho^2}\begin{bmatrix} 1 & -\rho & 0 & \cdots & 0 \\ -\rho & 1+\rho^2 & -\rho & \cdots & 0 \\ 0 & -\rho & 1+\rho^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

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

Computation Process

Step 1: Eigendecomposition of $\mathbf{Q}_1$

We first compute the eigendecomposition of $\mathbf{Q}_1$:

$$\mathbf{Q}_1 = \mathbf{V}\ \mathbf{V}^T$$

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

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

This means that the eigenvectors of $\mathbf{Q}$ are $\mathbf{v}_j \otimes \mathbf{v}_i$ and the corresponding eigenvalues are $\lambda_i + \lambda_j$.

Step 2: Computation of Marginal Standard Deviations

Using the eigendecomposition of $\mathbf{Q}_1$, we compute the marginal standard deviations and store them in a vector $$, i.e. $\sigma_i = \sqrt{\Sigma_{ii}}$:

$$= \sqrt{\sum_{i,j} \frac{(\mathbf{v}_j \otimes \mathbf{v}_i)^2}{(\lambda_i + \lambda_j)^\nu}}$$

where $\mathbf{v}_i$ are the eigenvectors and λ_i are the eigenvalues of $\mathbf{Q}_1$. Thus, σ will be a vector of length n^2.

Step 3: Scaling the Eigendecomposition

We scale the eigendecomposition of $\mathbf{Q}$ using the marginal standard deviations:

$$\begin{aligned}
\tilde{\mathbf{Q}} &= \mathbf{D}\mathbf{Q}\mathbf{D} \\
&= \mathbf{D}(\mathbf{V} \otimes \mathbf{V})(\ \otimes \mathbf{I} + \mathbf{I} \otimes\ )^\nu (\mathbf{V} \otimes \mathbf{V})^T \mathbf{D} \\
&= (\tilde{\mathbf{V}} \otimes \tilde{\mathbf{V}})(\tilde{}\ \otimes \mathbf{I} + \mathbf{I} \otimes \tilde{}\ )(\tilde{\mathbf{V}} \otimes \tilde{\mathbf{V}})^T
\end{aligned}$$

where $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sigma_i$.

Step 4: Efficient Computation of Log-Density

Using this scaled eigendecomposition, we efficiently compute:

1. Log-determinant: $\log|\tilde{\mathbf{Q}}| = \sum_{i,j} \log(\tilde{\lambda}_i + \tilde{\lambda}_j)$
2. Quadratic form: $\mathbf{z}^T \tilde{\mathbf{Q}} \mathbf{z} = \sum_{i,j} (\tilde{\lambda}_i + \tilde{\lambda}_j) y_{ij}^2$, where $y_{ij} = (\tilde{\mathbf{v}}_j \otimes \tilde{\mathbf{v}}_i)^T \mathbf{z}$

This approach allows for efficient computation of the Gaussian copula density without explicitly forming the full $n^2 \times n^2$ precision matrix $\mathbf{Q}$ or its inverse .

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:

$$\lambda_k = \sum_{j=0}^{n-1} c_j e^{-2\pi i j k/n}, \quad k = 0, 1, ..., n-1$$

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

$$Cv = \frac{1}{n}\text{DFT}(\text{DFT}(c) \odot \text{IDFT}(v))$$

3. The determinant of C is the product of its eigenvalues:

$$\det(C) = \prod_{k=0}^{n-1} \lambda_k$$

4. 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 $\mathbf{c}$:

$$\Lambda_{k,l} = \sum_{i=0}^{n-1} \sum_{j=0}^{N-1} c_{ij} e^{-2\pi i(ki/n + lj/N)}, \quad k = 0, \ldots, n-1, l = 0, \ldots, N-1$$

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

$$Cv = \text{IDFT2}(\text{DFT2}(c) \odot \text{DFT2}(v))$$

3. The determinant of C is the product of its eigenvalues:

$$\det(C) = \prod_{k=0}^{n-1} \prod_{l=0}^{N-1} \Lambda_{k,l}$$

Computational Advantages

The circulant structure allows for efficient computation using the Fast Fourier Transform (FFT):

1. Matrix-vector multiplication: For a circulant matrix C with base c and a vector v

$$Cv = \sqrt{n}\text{DFT}(\text{DFT}(c) \odot \text{IDFT}(v)),$$

2. Matrix inverse: The base of C^{-1} is given by

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

Approximations for Q1

Let Q_1 be the precision matrix of a one-dimensional AR(1) process with n observations. The exact form of Q_1 is:

$$\mathbf{Q}_1 = \frac{1}{1-\rho^2}\begin{bmatrix} 1 & -\rho & 0 & \cdots & 0 \\ -\rho & 1+\rho^2 & -\rho & \cdots & 0 \\ 0 & -\rho & 1+\rho^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Circulant Approximation

The circulant approximation to Q_1, denoted as $\mathbf{Q}_1^{(circ)}$, is:

$$\mathbf{Q}_1^{(circ)} = \frac{1}{1-\rho^2}\begin{bmatrix} 1+\rho^2 & -\rho & 0 & \cdots & 0 & -\rho \\ -\rho & 1+\rho^2 & -\rho & \cdots & 0 & 0 \\ 0 & -\rho & 1+\rho^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -\rho & 0 & 0 & \cdots & -\rho & 1+\rho^2 \end{bmatrix}$$

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

Folded Circulant Approximation

The folded circulant approximation, $\mathbf{Q}_1^{(fold)}$, 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:

$$\mathbf{Q}_1^{(fold)} = \frac{1}{1-\rho^2}\begin{bmatrix} 1-\rho+\rho^2 & -\rho & 0 & \cdots & 0 & 0 \\ -\rho & 1+\rho^2 & -\rho & \cdots & 0 & 0 \\ 0 & -\rho & 1+\rho^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\rho & 1-\rho+\rho^2 \end{bmatrix}$$

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 $n \times n$ grid, we construct the full precision matrix Q using a Kronecker sum:

$$\mathbf{Q} = (\mathbf{Q}_1 \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{Q}_1)^{(\nu+1)}$$

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

When we approximate Q_1 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:

$$\mathbf{Q} = \mathbf{Q}_1 \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{Q}_1$$

Now, let Q_1 be approximated by a circulant matrix C with base vector $c = [c, c_1, ..., c_{n-1}]$. Then:

$$\mathbf{Q}_1 \approx \mathbf{C} = \begin{pmatrix} c_0 & c_1 & \cdots & c_{n-1} \\ c_{n-1} & c_0 & \cdots & c_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & \cdots & c_0 \end{pmatrix}$$

The Kronecker product $C \otimes I$ results in a block matrix where each block is a scalar multiple of I:

$$\mathbf{C} \otimes \mathbf{I} = \begin{pmatrix} c_0\mathbf{I} & c_1\mathbf{I} & \cdots & c_{n-1}\mathbf{I} \\ c_{n-1}\mathbf{I} & c_0\mathbf{I} & \cdots & c_{n-2}\mathbf{I} \\ \vdots & \vdots & \ddots & \vdots \\ c_1\mathbf{I} & c_2\mathbf{I} & \cdots & c_0\mathbf{I} \end{pmatrix}$$

Similarly, I C results in a block matrix where each block is a copy of C:

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

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

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

where each $\mathbf{B_i}$ is a circulant matrix. Specifically, $\mathbf{B_0} = c_0 I + C$, and for $i \neq 0$, $\mathbf{B_i} = c_i I$.

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

$$\mathbf{c} = \begin{bmatrix} 2+2\rho^2 & -\rho & 0 & \cdots & -\rho \\ -\rho & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\rho & 0 & 0 & \cdots & 0 \end{bmatrix}$$

This base matrix c captures the structure of the precision matrix 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 c for our BCCB approximation of Q. For an n $\times$ n grid, c is an n $\times$ n matrix:

$$\mathbf{c} = \begin{bmatrix} 2+2\rho^2 & -\rho & 0 & \cdots & 0 & -\rho \\ -\rho & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -\rho & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

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

2. Compute Initial Eigenvalues

We compute the initial eigenvalues of Q using the 2D Fast Fourier Transform (FFT) of 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 with a constant diagonal. 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 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: $\tilde{\Lambda} = \sigma^2 \Lambda$

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 $\tilde{\mathbf{Q}}$ can be efficiently calculated as the sum of the logarithms of the scaled eigenvalues:

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

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: $\hat{\mathbf{y}} = \tilde{\Lambda} \odot \hat{\mathbf{z}}$

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 Q_1 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:

$$\begin{bmatrix} x_{11} & \cdots & x_{1n} & x_{1n} & \cdots & x_{11} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nn} & x_{nn} & \cdots & x_{n1} \\ x_{n1} & \cdots & x_{nn} & x_{nn} & \cdots & x_{n1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{11} & \cdots & x_{1n} & x_{1n} & \cdots & x_{11} \end{bmatrix}$$

This reflection creates a 2n $\times$ 2n matrix. This data is then stacked column-wise 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.

Table 1: Table 1. Benchmarking how long it takes to evaluate the density of a Mátern(ν)-like field with correlation parameter ρ, scaled to have unit marginal variance.

Q_size	Exact	Circulant		Folded	
	Time	Time	Speed-Up	Time	Speed-Up
100x100	194.69 s	33.89 s	5.75x	44.77 s	4.35x
400x400	309.61 s	37.45 s	8.27x	122.78 s	2.52x
900x900	790.30 s	85.65 s	9.23x	155.47 s	5.08x
1600x1600	1.96ms	98.71 s	19.86x	243.64 s	8.05x
3600x3600	8.66ms	151.70 s	57.1x	484.07 s	17.89x
10000x10000	71.81ms	343.07 s	209.31x	1.36ms	52.76x
19600x19600	246.40ms	667.58 s	369.09x	2.89ms	85.32x
40000x40000	997.32ms	1.44ms	694.46x	7.69ms	129.69x

The results demonstrate significant computational gains from both approximation methods, with the efficiency advantage increasing for larger grid sizes. Key observations include:

1. **Scalability**: Both approximation methods show superior scalability compared to the exact method. As the grid size increases, the speed-up factor generally increases, indicating that the approximations become increasingly advantageous for larger spatial

fields.

2. **Circulant Approximation Performance**: The circulant approximation consistently outperforms the folded circulant approximation in terms of speed. For the largest grid size (40000x40000), it achieves a remarkable 694.46x speed-up over the exact method.

3. **Folded Circulant Approximation**: While not as fast as the circulant approximation, the folded circulant method still offers substantial speed improvements, reaching a 129.69x speed-up for the 40000x40000 grid.

4. **Trade-off Considerations**: The choice between the circulant and folded circulant approximations may depend on the specific requirements of the application. While the circulant approximation is faster, the folded circulant method may offer better accuracy, particularly near the edges of the spatial field.

These results underscore the practical value of our approximation methods, especially for large-scale spatial analyses where computational efficiency is crucial. The substantial speed-ups achieved, particularly for larger grids, demonstrate the potential of these methods to enable the analysis of much larger spatial datasets than previously feasible with exact methods.