



**HAL**  
open science

# ParaCircE, a parallel Gaussian Random Field (GRF) generator

Simon Legrand, Géraldine Pichot

► **To cite this version:**

Simon Legrand, Géraldine Pichot. ParaCircE, a parallel Gaussian Random Field (GRF) generator. GS21 - SIAM Conference on Mathematical and Computational Issues in the Geosciences, Jun 2021, Milan / Virtual, Italy. hal-03520463

**HAL Id: hal-03520463**

**<https://inria.hal.science/hal-03520463>**

Submitted on 11 Jan 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# ParaCircE, a parallel Gaussian Random Field (GRF) generator

Simon Legrand<sup>1</sup>    Géraldine Pichot<sup>1</sup>

<sup>1</sup>Inria Paris and ENPC

SIAM GS21

## 1 Introduction

- Gaussian Random Fields
- General algorithm

## 2 ParaCircE

- Circulant embedding algorithm
- Features
- Technical overview
- API

## 3 Benchmarks

- Tests environment
- Strong scaling
- Memory scaling

## 4 Conclusion

# Contents

- 1 Introduction
  - Gaussian Random Fields
  - General algorithm

- 2 ParaCircE
  - Circulant embedding algorithm
  - Features
  - Technical overview
  - API

- 3 Benchmarks
  - Tests environment
  - Strong scaling
  - Memory scaling

- 4 Conclusion

# Gaussian Random Fields

## Context

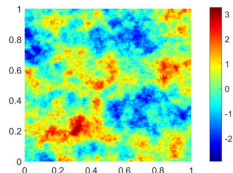
In hydrogeology, permeability/transmissivity are often modeled by **second order stationary fields**, with **lognormal** distributions.

Classical applications are:

- Groundwater resources management
- Pollution propagation studies
- Waste storage studies in deep geological media



Sand and gravel deposits in Switzerland. Gelhar (1993)



Matérn isotropic covariance,  $\nu = 1$ ,  $n = 128^2$ ,  $\lambda = 0.125$

# Contents

## 1 Introduction

- Gaussian Random Fields
- **General algorithm**

## 2 ParaCircE

- Circulant embedding algorithm
- Features
- Technical overview
- API

## 3 Benchmarks

- Tests environment
- Strong scaling
- Memory scaling

## 4 Conclusion

# Gaussian Random Fields

## 1D General algorithm

Let  $\Omega \in \mathbb{R}$ . To generate  $\mathbf{y}$ , a Gaussian vector, with zero mean and covariance matrix  $\mathbf{R}$ , defined on a  $N + 1$  points regular grid of  $\Omega$ :

- 1 Factorize  $\mathbf{R} = \mathbf{B}\mathbf{B}^T$
- 2 Generate  $\boldsymbol{\theta} = (\theta_0, \dots, \theta_N)^T$ , a realization of standard normal variables.
- 3 One realization is obtained with  $\mathbf{y} = \mathbf{B}\boldsymbol{\theta}$

## Among existing methods

- Cholesky factorization, efficient on small problems, but  $O(N^3)$
- Karhunen Loève series expansion, expensive when  $\lambda \ll L$
- H-Matrices [Feischl et al., 2018], suited for unstructured meshes
- **Circulant Embedding** [Dietrich et al., 1993], [Graham et al., 2018], with exact correlation structure, suited for large regular grids.

# Gaussian Random Fields

## 1D General algorithm

Let  $\Omega \in \mathbb{R}$ . To generate  $\mathbf{y}$ , a Gaussian vector, with zero mean and covariance matrix  $\mathbf{R}$ , defined on a  $N + 1$  points regular grid of  $\Omega$ :

- 1 Factorize  $\mathbf{R} = \mathbf{B}\mathbf{B}^T$
- 2 Generate  $\boldsymbol{\theta} = (\theta_0, \dots, \theta_N)^T$ , a realization of standard normal variables.
- 3 One realization is obtained with  $\mathbf{y} = \mathbf{B}\boldsymbol{\theta}$

## Among existing methods

- Cholesky factorization, efficient on small problems, but  $O(N^3)$
- Karhunen Loève series expansion, expensive when  $\lambda \ll L$
- H-Matrices [Feischl et al., 2018], suited for unstructured meshes
- **Circulant Embedding** [Dietrich et al., 1993], [Graham et al., 2018], with exact correlation structure, suited for large regular grids.



# Contents

- 1 Introduction
  - Gaussian Random Fields
  - General algorithm
- 2 ParaCircE
  - Circulant embedding algorithm
  - Features
  - Technical overview
  - API
- 3 Benchmarks
  - Tests environment
  - Strong scaling
  - Memory scaling
- 4 Conclusion

# 1D Circulant Embedding algorithm

- 1 Sample the covariance function to get the vector  $\mathbf{a} = (c_0, \dots, c_N, c_{N-1}, \dots, c_1) \in \mathbb{R}^{2N}$ , first row of a circulant matrix  $\mathbf{A}$ .
- 2 If  $N$  is large enough,  $\mathbf{A}$  is SPD, we compute its eigenvalues with FFT, otherwise increase  $N$  (padding).  
 $\mathbf{s} = \mathbf{F}\mathbf{a}$  and set  $\mathbf{D} = \text{Diag}(\mathbf{s})$
- 3 Generate realizations of standard normal vectors of size  $2N$   
 $\theta = \theta^{re} + i\theta^{im}$
- 4 Apply iFFT to compute  $\mathbf{C}^*\theta$ , with  $\mathbf{C}^* = \frac{1}{\sqrt{2N}}\mathbf{F}^*\mathbf{D}^{\frac{1}{2}}$
- 5 Retrieve two GRF  
 $\mathbf{Y}_1 = (\mathbf{C}^*\theta)^{re}(0 : N), \quad \mathbf{Y}_2 = (\mathbf{C}^*\theta)^{im}(0 : N)$

# 1D Circulant Embedding algorithm

- 1 Sample the covariance function to get the vector  $\mathbf{a} = (c_0, \dots, c_N, c_{N-1}, \dots, c_1) \in \mathbb{R}^{2N}$ , first row of a circulant matrix  $\mathbf{A}$ .
- 2 If  $N$  is large enough,  $\mathbf{A}$  is SPD, we compute its eigenvalues with FFT, otherwise increase  $N$  (padding).  
 $\mathbf{s} = \mathbf{F}\mathbf{a}$  and set  $\mathbf{D} = \text{Diag}(\mathbf{s})$
- 3 Generate realizations of standard normal vectors of size  $2N$   
 $\theta = \theta^{re} + i\theta^{im}$
- 4 Apply iFFT to compute  $\mathbf{C}^*\theta$ , with  $\mathbf{C}^* = \frac{1}{\sqrt{2N}}\mathbf{F}^*\mathbf{D}^{\frac{1}{2}}$
- 5 Retrieve two GRF  
 $\mathbf{Y}_1 = (\mathbf{C}^*\theta)^{re}(0:N), \quad \mathbf{Y}_2 = (\mathbf{C}^*\theta)^{im}(0:N)$

# 1D Circulant Embedding algorithm

- 1 Sample the covariance function to get the vector  $\mathbf{a} = (c_0, \dots, c_N, c_{N-1}, \dots, c_1) \in \mathbb{R}^{2N}$ , first row of a circulant matrix  $\mathbf{A}$ .
- 2 If  $N$  is large enough,  $\mathbf{A}$  is SPD, we compute its eigenvalues with FFT, otherwise increase  $N$  (padding).  
 $\mathbf{s} = \mathbf{F}\mathbf{a}$  and set  $\mathbf{D} = \text{Diag}(\mathbf{s})$
- 3 Generate realizations of standard normal vectors of size  $2N$   
 $\boldsymbol{\theta} = \boldsymbol{\theta}^{re} + i\boldsymbol{\theta}^{im}$
- 4 Apply iFFT to compute  $\mathbf{C}^*\boldsymbol{\theta}$ , with  $\mathbf{C}^* = \frac{1}{\sqrt{2N}}\mathbf{F}^*\mathbf{D}^{\frac{1}{2}}$
- 5 Retrieve two GRF  
 $\mathbf{Y}_1 = (\mathbf{C}^*\boldsymbol{\theta})^{re}(0 : N), \quad \mathbf{Y}_2 = (\mathbf{C}^*\boldsymbol{\theta})^{im}(0 : N)$

# 1D Circulant Embedding algorithm

- 1 Sample the covariance function to get the vector  $\mathbf{a} = (c_0, \dots, c_N, c_{N-1}, \dots, c_1) \in \mathbb{R}^{2N}$ , first row of a circulant matrix  $\mathbf{A}$ .
- 2 If  $N$  is large enough,  $\mathbf{A}$  is SPD, we compute its eigenvalues with FFT, otherwise increase  $N$  (padding).  
 $\mathbf{s} = \mathbf{F}\mathbf{a}$  and set  $\mathbf{D} = \text{Diag}(\mathbf{s})$
- 3 Generate realizations of standard normal vectors of size  $2N$   
 $\boldsymbol{\theta} = \boldsymbol{\theta}^{re} + i\boldsymbol{\theta}^{im}$
- 4 Apply iFFT to compute  $\mathbf{C}^*\boldsymbol{\theta}$ , with  $\mathbf{C}^* = \frac{1}{\sqrt{2N}}\mathbf{F}^*\mathbf{D}^{\frac{1}{2}}$
- 5 Retrieve two GRF  
 $\mathbf{Y}_1 = (\mathbf{C}^*\boldsymbol{\theta})^{re}(0:N), \quad \mathbf{Y}_2 = (\mathbf{C}^*\boldsymbol{\theta})^{im}(0:N)$

# 1D Circulant Embedding algorithm

- 1 Sample the covariance function to get the vector  $\mathbf{a} = (c_0, \dots, c_N, c_{N-1}, \dots, c_1) \in \mathbb{R}^{2N}$ , first row of a circulant matrix  $\mathbf{A}$ .
- 2 If  $N$  is large enough,  $\mathbf{A}$  is SPD, we compute its eigenvalues with FFT, otherwise increase  $N$  (padding).  
 $\mathbf{s} = \mathbf{F}\mathbf{a}$  and set  $\mathbf{D} = \text{Diag}(\mathbf{s})$
- 3 Generate realizations of standard normal vectors of size  $2N$   
 $\boldsymbol{\theta} = \boldsymbol{\theta}^{re} + i\boldsymbol{\theta}^{im}$
- 4 Apply iFFT to compute  $\mathbf{C}^*\boldsymbol{\theta}$ , with  $\mathbf{C}^* = \frac{1}{\sqrt{2N}}\mathbf{F}^*\mathbf{D}^{\frac{1}{2}}$
- 5 Retrieve two GRF  
 $\mathbf{Y}_1 = (\mathbf{C}^*\boldsymbol{\theta})^{re}(0 : N), \quad \mathbf{Y}_2 = (\mathbf{C}^*\boldsymbol{\theta})^{im}(0 : N)$

# Contents

- 1 Introduction
  - Gaussian Random Fields
  - General algorithm
- 2 ParaCircE
  - Circulant embedding algorithm
  - **Features**
  - Technical overview
  - API
- 3 Benchmarks
  - Tests environment
  - Strong scaling
  - Memory scaling
- 4 Conclusion

# Features

- **Matérn** family of covariance (example in 2D)

$$\rho_{2D}(\mathbf{x}, \nu, \boldsymbol{\lambda}) = \kappa\left(\frac{\sqrt{\lambda_2^2 x^2 + \lambda_1^2 y^2}}{\lambda_1 \lambda_2}, \nu\right),$$

with

$$\kappa(r, \nu) = \frac{2^{1-\nu}}{\Gamma(\nu)} (\sqrt{2\nu} r)^\nu K_\nu(\sqrt{2\nu} r),$$

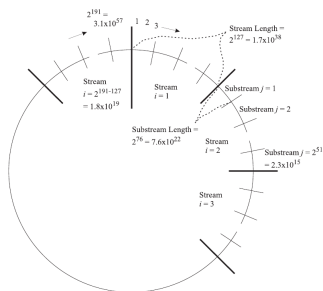
- $\mathbf{x} = (x, y)$
  - $\Gamma$  the gamma function
  - $K_\nu$  the modified Bessel function of the second kind
  - $\lambda_i$  the correlation length along direction  $i$
  - $\nu > 0$  a smoothness parameter.
- 
- **Parallel** (MPI)  
One direction parallelism (FFTW). The reproducibility is guaranteed whatever the number of processes (RngStream).



# Features

- **Independance** and **reproducibility** thanks to the RngStream library [L'Ecuyer et al., 2002]

$2^{191}$ -periodic set of pseudo random numbers, divided into **Streams** and **Substreams**.



- **Accelerated** minimum domain size computation

If padding is required, save computational resources by approximating the required domain size to get **positive eigenvalues** [Graham et al., 2018, Legrand et al., 2021].

# Contents

- 1 Introduction
  - Gaussian Random Fields
  - General algorithm
- 2 ParaCircE
  - Circulant embedding algorithm
  - Features
  - **Technical overview**
  - API
- 3 Benchmarks
  - Tests environment
  - Strong scaling
  - Memory scaling
- 4 Conclusion

- C++17/MPI **headers** (templated) library
  - Static dimension 2D/3D
  - Variable numerical precision (double/long double)
- Dependencies
  - MPI
  - FFTW\_MPI [Frigo et al., 2005]
  - RngStream [L'Ecuyer et al., 2002]
- Installation with **CMake**
  - Portability
  - Easy dependencies management

# Contents

- 1 Introduction
  - Gaussian Random Fields
  - General algorithm
- 2 ParaCircE
  - Circulant embedding algorithm
  - Features
  - Technical overview
  - **API**
- 3 Benchmarks
  - Tests environment
  - Strong scaling
  - Memory scaling
- 4 Conclusion

## Data structure instantiation

- Domain

```
DiscretDomain2D<double> d{  
    {{lx_min, lx_max, nx}, {ly_min, ly_max, ny}}  
};
```

- Correlation function

```
auto f = create_function2D<double>(correl_type,  
                                   lcx, lcy);
```

- Field generator

```
auto generator = GRF(MPI_COMM_WORLD, d, f);
```

## Field generation

```
auto field = generator.generate(nx_local);
```

## Data structure instantiation

- Domain

```
DiscretDomain2D<double> d{  
    {{lx_min, lx_max, nx}, {ly_min, ly_max, ny}}  
};
```

- Correlation function

```
auto f = create_function2D<double>(correl_type,  
                                   lcx, lcy);
```

- Field generator

```
auto generator = GRF(MPI_COMM_WORLD, d, f);
```

## Field generation

```
auto field = generator.generate(nx_local);
```

## Data structure instantiation

- Domain

```
DiscretDomain2D<double> d{  
    {{lx_min, lx_max, nx}, {ly_min, ly_max, ny}}  
};
```

- Correlation function

```
auto f = create_function2D<double>(correl_type,  
                                   lcx, lcy);
```

- Field generator

```
auto generator = GRF(MPI_COMM_WORLD, d, f);
```

## Field generation

```
auto field = generator.generate(nx_local);
```

## Data structure instantiation

- Domain

```
DiscretDomain2D<double> d{  
    {{lx_min, lx_max, nx}, {ly_min, ly_max, ny}}  
};
```

- Correlation function

```
auto f = create_function2D<double>(correl_type,  
                                   lcx, lcy);
```

- Field generator

```
auto generator = GRF(MPI_COMM_WORLD, d, f);
```

## Field generation

```
auto field = generator.generate(nx_local);
```



For **Monte-Carlo** use, change stream/substream of the pseudo RNG RngStream.

```
//generator.set_state(stream, substream);
```

## Field generation

```
...  
for (size_t i=0; i<nb_mc; i++)  
{  
    generator.set_state(0, i);  
    auto field = generator.generate(nx_local);  
    ...  
}
```

# Contents

- 1 Introduction
  - Gaussian Random Fields
  - General algorithm
- 2 ParaCircE
  - Circulant embedding algorithm
  - Features
  - Technical overview
  - API
- 3 **Benchmarks**
  - **Tests environment**
  - Strong scaling
  - Memory scaling
- 4 Conclusion

# Tests environment

- Instrumentation performed with **Score-P/Scalasca** tools
- Compilation options: `-O3`
- Execution on Inria Paris CLEPS cluster
  - Xeon 5218, 2.4GHz
  - 192Go RAM 2667MHz
  - 100Gb/s Infiniband network
- 1 core/MPI process
- **Exclusive** access to the nodes
- No hyperthreading

# Contents

- 1 Introduction
  - Gaussian Random Fields
  - General algorithm
- 2 ParaCircE
  - Circulant embedding algorithm
  - Features
  - Technical overview
  - API
- 3 **Benchmarks**
  - Tests environment
  - **Strong scaling**
  - Memory scaling
- 4 Conclusion

# Strong scaling

- **Fixed** size problem  
(2048\*512\*512)
- **Increasing number** of processes

Maximum acceleration given by Amdhal law:

$$A(N) = \frac{1}{s + \frac{p}{N}}$$

$N$ : Number of tasks

$s$ : sequential time fraction = 0.014

$p$ : parallel time fraction = 0.986

Total time  $\approx$  41min

## Results

- Scales like FFTW

# Strong scaling

- **Fixed** size problem  
(2048\*512\*512)
- **Increasing number** of processes

Maximum acceleration given by Amdhal law:

$$A(N) = \frac{1}{s + \frac{p}{N}}$$

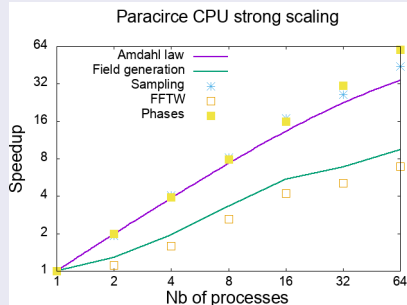
$N$ : Number of tasks

$s$ : sequential time fraction = 0.014

$p$ : parallel time fraction = 0.986

Total time  $\approx$  41min

## Results



- Scales like **FFTW**

# Strong scaling

- **Fixed** size problem  
(2048\*512\*512)
- **Increasing number** of processes

Maximum acceleration given by Amdhal law:

$$A(N) = \frac{1}{s + \frac{p}{N}}$$

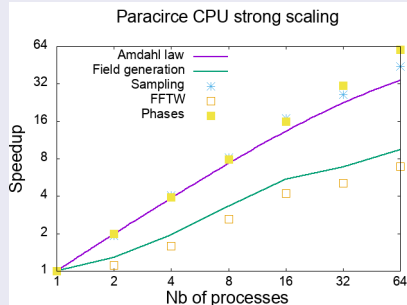
$N$ : Number of tasks

$s$ : sequential time fraction = 0.014

$p$ : parallel time fraction = 0.986

Total time  $\approx$  41min

## Results



- Scales like **FFTW**

# Contents

- 1 Introduction
  - Gaussian Random Fields
  - General algorithm
- 2 ParaCircE
  - Circulant embedding algorithm
  - Features
  - Technical overview
  - API
- 3 **Benchmarks**
  - Tests environment
  - Strong scaling
  - **Memory scaling**
- 4 Conclusion



# Memory scaling

- The goal of ParaCircE is to generate **large** fields.
- Main constraint: **Memory occupation** (**R**esident **S**et **S**ize)

**Circulant Embedding** algorithm implies **symmetrizing** the domain.  
In dimension  $d$ :

$$RSS \approx complexSize * (2N)^d$$

For  $N = 512$ ,  $d = 3$ , with a complex size of 128 bits:

$$RSS \approx 16 * 8 * 512^3 \approx 16GB$$

# Memory scaling

- The goal of ParaCircE is to generate **large** fields.
- Main constraint: **Memory occupation** (**R**esident **S**et **S**ize)

**Circulant Embedding** algorithm implies **symmetrizing** the domain.  
In dimension  $d$ :

$$RSS \approx complexSize * (2N)^d$$

For  $N = 512$ ,  $d = 3$ , with a complex size of 128 bits:

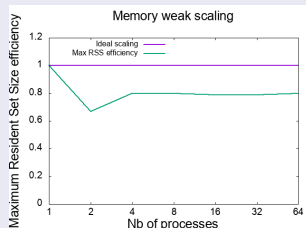
$$RSS \approx 16 * 8 * 512^3 \approx 16GB$$

# Memory weak scaling

- Both problem size and number of tasks increase
- **Constant workload** per task,  $\approx 512^3$  samples

## Results

Nb_proc	Time(min)	Max RSS(GB)	RSS Efficiency
1	7.8	15.9	1.00
2	11.2	23.9	0.67
4	16.8	20.0	0.80
8	23.2	20.0	0.80
16	31.0	20.1	0.79
32	41.4	20.1	0.79
64	70.3	20.0	0.80



Relatively **low** execution time

- $\sim$  **134 millions** elements: 1 proc  $\rightarrow$  **7min48s**
- $\sim$  **8.6 billions** elements: 64 proc  $\rightarrow$  **1h10min** (would require 1TB in sequential)

**Constant** memory scaling!

# Conclusion

- **Easy** to install and use
- **Fast** even for very large cases
- **Accelerated** method for automated padding computation

## Perspectives

- Periodic field generation
- Multithread
- What do you need?

**Gitlab repository:** <https://gitlab.inria.fr/slegrand/paracirce>

# Conclusion

- **Easy** to install and use
- **Fast** even for very large cases
- **Accelerated** method for automated padding computation

## Perspectives

- Periodic field generation
- Multithread
- What do you need?

**Gitlab repository:** <https://gitlab.inria.fr/slegrand/paracirce>

# For Further Reading I



LEcuyer, Pierre and Simard, Richard and Chen, E. and Kelton, David

An Object-Oriented Random-Number Package With Many Long Streams And Substreams

Operations Research, 2002, Vol. 50, pp. 1073-1075



I. G. Graham, F. Y. Kuo, D. Nuyens, R. Scheichl, and I. H. Sloan  
*Analysis of Circulant Embedding Methods for Sampling Stationary Random Fields*


SIAM Journal on Numerical Analysis, 2018, Vol. 56, No. 3 : pp. 1871-1895





Simon Legrand, Geraldine Pichot, Nathanael Tepakbong  
*Algorithms to speed up the generation of stationary Gaussian Random Fields with the Circulant Embedding method*

HAL 2021

# For Further Reading II

 M. Frigo and S. G. Johnson.  
The design and implementation of fftw3.  
*Proc. IEEE* 93, 2:216–231, 2005.

 M. Feischl, F.Y. Kuo, and I.H. Sloan.  
Fast random field generation with h-matrices.  
*Numer. Math.*, 140:639676, 2018.

 C. R. Dietrich and G. N. Newsam.  
A fast and exact method for multidimensional gaussian stochastic simulations.  
*Water Resources Research*, 29(8):2861–2869, 1993.