A Python Library for simulating Gaussian random fields

Khaya Donald Mpehle

Here we will introduce a library of python functions, PyGRyF, concerned with simulating Gaussian Random fields. A random field(RF) is a random function $F(t,\omega) \equiv F_t$, in which the t is called the index and is a single or multi-dimensional quantity. A random field F_t therefore assigns a random variable to each point t in the index space T. If t is a one-dimensional index, then a random field reduces to a stochastic process(SP). The variations of topography would be modelled by a random field H_t on a two-dimensional domain $t \in \mathbb{R}^2$; Random fluctuations of the wind field across the Earth's surface would be modelled be a random field with its index set a subset of the surface of a sphere. In general then a random field is an extension of a stochastic process, allowing the index space T to be a manifold. A Gaussian random field X_t is one that has a Gaussian distribution for each $t \in T$.

It is our goal to produce a Python library for the numerical simulation of Gaussian random fields in one-dimension and two-dimensional rectangular domains using two well established numerical techniques. The first is via truncation of the Karhunen-Loève(KL) expansion of the RF.

1 The Karhunen Loève approximation

Let X_t be a zero mean Gaussian field, where $t \in T$, with covariance function

$$R(t,s) = E\left[X_t X_s\right]. \tag{1}$$

The Karhunen-Loève expansion theorem states that X_t can be expanded as

$$X_t = \sum_{k=1}^{\infty} \sqrt{\lambda_k} Z_k \phi_k(t). \tag{2}$$

In this expansion Z_k are iid standard normal variables while ϕ_k and λ_k are respectively the eigen-functions and -values of the RF's covariance function. Therefore ϕ_k and λ_k satisfy the eigenvalue problem

$$\int_{t\in T} R(t,s)\phi_k(s)ds = \lambda_k \phi_k(t) \tag{3}$$

The Karhunen-Loève expansion can roughly be thought of as "separating" the deterministic and random components of the Gaussian RF.

A means of approximating the RF X_t is to truncate the KL expansion after N terms. This is termed the Karhunen-Loève approximation of order N. With a finite number of terms in the expansion, the only remaining task is to find the eigenvalues and -functions of the convariance

function. There are several ways to do this, here we shall focus on using the Nyström method. The integral eigenvalue problem (3) is approximated by an M-point quadrature $\{w_i, x_i\}_{i=1}^M$, the resulting expression is considered on each of the quadrature points:

$$\sum_{i=1}^{M} w_i R(t_j, s_i)\phi_k(s_i) = \lambda_k \phi(t_j), \qquad j = 0, 1, 2, \ldots, M. \tag{4}$$

The discrete expression (4) reduces to the matrix eigenvalue problem

$$\boldsymbol{C}\mathbf{W}\boldsymbol{\phi}_\mathbf{k} = \lambda_\mathbf{k}\boldsymbol{\phi}_\mathbf{k} \tag{5}$$

where we have introduced the diagonal matrix of weights $\boldsymbol{W} = \text{diag}\{w_1, \ldots, w_M\}$ and the covariance matrix $(\boldsymbol{C})_{ij} = R(t_i, t_j)$. Solving this eigenvalue problem will determine the eigenvectors and -values of the discretised system of (3) and allow the determination of the discrete KL approximation. We note as a practical consideration that (5) can be modified by left multiplying both sides by $\boldsymbol{W}^{1/2}$, such that the system

$$\boldsymbol{W}^{1/2}\boldsymbol{C}\boldsymbol{W}^{1/2}\boldsymbol{y}_k = \lambda_k \boldsymbol{y}_k, \tag{6}$$

where $\boldsymbol{y}_k = \boldsymbol{W}^{1/2}\boldsymbol{\phi}_k$, is obtained. This system is advantageous as $\boldsymbol{W}^{1/2}\boldsymbol{C}\boldsymbol{W}^{1/2}$ is a positive semi-definite symmetric matrix and therefore has orthogonal eigenvectors with positive,real eigen values.

1.1 Examples

We have produced code using python and its numpy library to compute the Karhunen-Loève expansion on one- and two-dimensional rectangular domains.So far the only quadrature method we have implemented is the "EOLE" method in which the quadrature points x_i are evenly spaced and the weights $w_i = |\Omega|/N$ are equal. We wish to extend the methodology to include multiple quadrature methods.

The function `KL_1DNys` will return a simulation of the Gaussian process X_t when given its covariance function $R(t, s)$. In figure 1, the eigenvalues of the Brownian motion process along with the fifth eigenfunction are compared to our EOLE Nyström simulation. One can see good agreement between the exact eigenvalues and our numerical approximation.

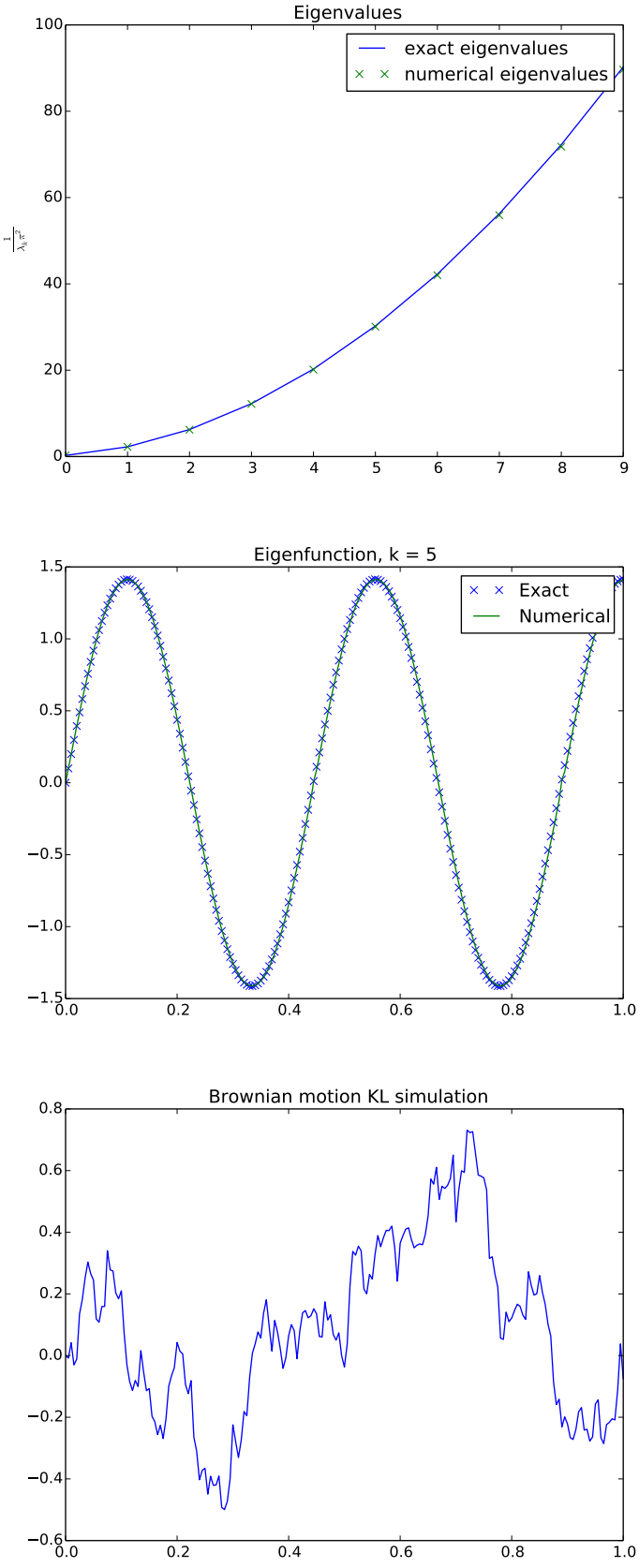

Figure 1: Top: Comparison of the numerical eigenvalues to the exact eigenvalues in the Brownian motion Karhunen-Loève function expansion.Middle: Comparison of the exact and numerical 5th eigenfunction. Bottom: The approximate Brownian motion in an $N = 200$ term expansion. Here M=200 quadrature points were used.

The function `KL_2DNys` will return a simulation of the Gaussian field X_t, $t \in \mathbb{R}$ with supplied covariance function, on a rectangular domain. As an example, consider the Gaussian RF with the stationary, isotropic exponential covariance given by

$$R(s,t) = \exp\left(-\frac{||s-t||}{\rho}\right) \tag{7}$$

where ρ is some scale radius. In figure 2, we see a plot of the eigenvalues of this covariance matrix along with the first 5 eigenfunctions, simulated on the domain $[0,1]\times[0,1]$ with 50×50 points. Note that this is a small number of simulation points to be using, but this is all that the the relatively weak computer's RAM allows to be used. Due to this limitation, The accompanying random field simulation in figure 2 is generated by first interpolating the associated eigenfunctions onto a finer grid. The decay of the eigenvalues and the form of the eigenfunctions look to be in agreement with implementations with other authors[source PieterRobJan], although a full convergence test with finer discretisations, on a more powerful computer, is called for.

2 Circulant Embedding methods

Suppose X_t is a stationary Gaussian random field so that its covariance function is of the form $R(s,t) = R(s-t)$. In such a case, it may be preferable to use the *Circulant Embedding Method*. The method is so-called because it exploits the fact that the covariance matrix of stationary SPs can be embedded into a larger circulant matrix. Then one can use the Fast Fourier Transform to compute the eigenvalues of the circulant matrix, and from there go on to simulate the RF. The details of the method are described in [reference Newsam and Dietrich, Woods and Chan]. What is desirable about the Circulant Embedding algorithm is its generation of a sample that has the exact covariance structure, and its speed. The function `circ_embed1D.py` will return an array containing the simulated Gaussian process when given a power of two `g`, the end points `a, b` of the domain and the covariance function. The sample size will be $N = 2^g$, as the Circulant Embedding method requires the sample size to be a power of two to be efficient. In figure 3 we plot a realisation of the SP with the exponential covariance function

$$R(s,t) = R(s-t) = \exp\left(\frac{|s-t|}{l}\right). \tag{8}$$

A similar function `circ_embed2D.py` implements the method in two-dimensions. As an example, we take an example given in Newsam and Dietrich, the Gaussian RF with the covariance

$$R(s,t) = \exp\left((s-t)^T A(s-t)\right) \tag{9}$$

where A is the positive-definite, symmetric matrix

$$A = \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}.$$

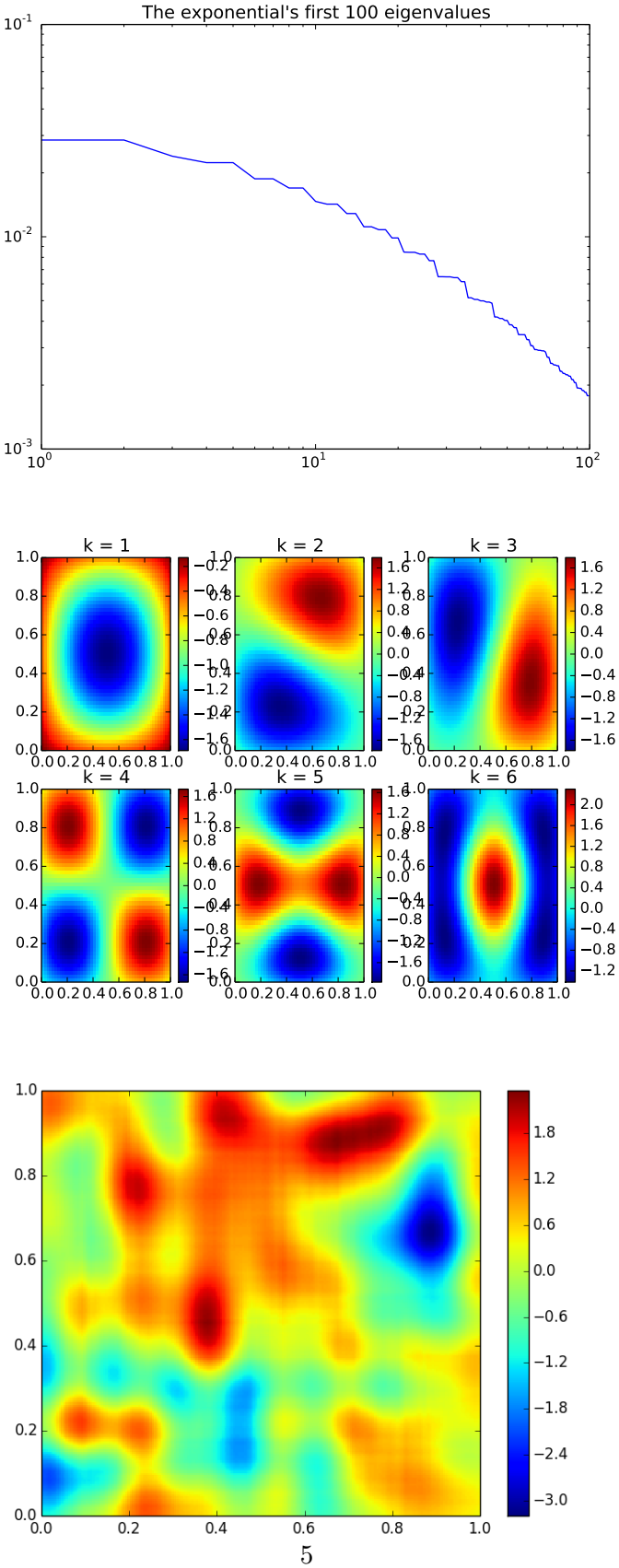

Figure 2: Top: Eigenvalues of the 2D exponential covariance function Gaussian RF. Middle: The first 6 eigenfunctions. Bottom: The random field realisation, using interpolated eigenfunctions. There are 50×50 points and the order of the expansion is $N = 100$.

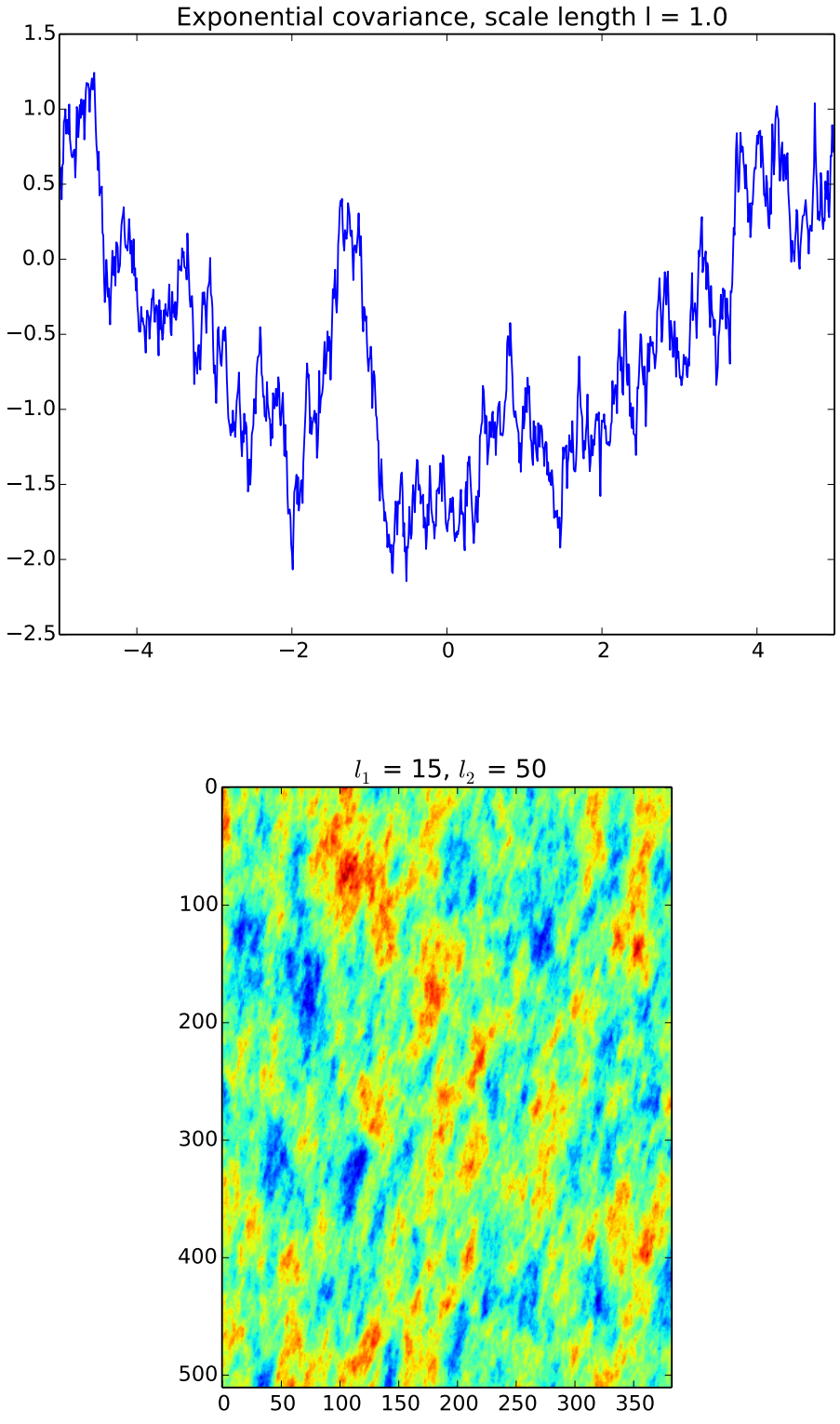

Figure 3: Top: A Realisation of the 1-D exponential random process.Bottom: A realisation of the Gaussian RF with anisotropic covariance(9).