**Generating a Correlated Weibull Random Field in MATLAB**

To generate a spatially correlated random field following a Weibull distribution with a given correlation length in MATLAB, you can follow these main steps:

1. **Define the Spatial Grid**: Set up a spatial grid where you want to generate the random field.
2. **Compute the Covariance Matrix**: Use a spatial correlation function (e.g., exponential decay) to compute the covariance matrix based on the correlation length.
3. **Generate a Gaussian Random Field**: Create a Gaussian random field with the specified spatial correlation using methods like Cholesky decomposition.
4. **Transform to Weibull Distribution**: Convert the Gaussian field to a Weibull-distributed field using the inverse cumulative distribution function (CDF) transformation.

### **Detailed Explanation and MATLAB Code**

#### **1. Define the Spatial Grid**

Set up a 2D grid of points where the random field will be generated.

matlab

Copy code

% Define grid size

n = 50; % Number of grid points in each dimension

[x, y] = meshgrid(linspace(0, 1, n), linspace(0, 1, n));

grid\_points = [x(:), y(:)]; % Flatten the grid into a list of points

#### **2. Compute the Covariance Matrix**

Compute the covariance matrix using an exponential decay function to introduce spatial correlation based on the correlation length.

matlab

Copy code

% Define correlation length

correlation\_length = 0.1;

% Number of points

num\_points = size(grid\_points, 1);

% Initialize covariance matrix

covariance\_matrix = zeros(num\_points);

% Compute pairwise distances and covariance

for i = 1:num\_points

for j = i:num\_points % Compute only upper triangle (symmetry)

distance = norm(grid\_points(i,:) - grid\_points(j,:));

covariance = exp(-distance / correlation\_length);

covariance\_matrix(i,j) = covariance;

covariance\_matrix(j,i) = covariance; % Symmetric matrix

end

end

#### **3. Generate a Gaussian Random Field**

Use Cholesky decomposition to generate a correlated Gaussian random field.

matlab

Copy code

% Cholesky decomposition of the covariance matrix

L = chol(covariance\_matrix, 'lower');

% Generate uncorrelated standard normal random variables

z = randn(num\_points, 1);

% Generate correlated Gaussian random field

gaussian\_field = L \* z;

% Reshape to 2D grid for visualization

gaussian\_field\_grid = reshape(gaussian\_field, n, n);

#### **4. Transform to Weibull Distribution**

Transform the Gaussian random field to a Weibull-distributed field using the inverse CDF (quantile function) of the Weibull distribution.

matlab

Copy code

% Weibull distribution parameters

shape\_param = 2.0; % k (shape parameter)

scale\_param = 1.0; % lambda (scale parameter)

% Convert Gaussian values to uniform values using the standard normal CDF

uniform\_values = normcdf(gaussian\_field);

% Apply the inverse Weibull CDF (quantile function)

weibull\_field = scale\_param \* (-log(1 - uniform\_values)).^(1 / shape\_param);

% Reshape to 2D grid for visualization

weibull\_field\_grid = reshape(weibull\_field, n, n);

#### **5. Visualize the Results**

Plot the generated Gaussian and Weibull random fields to visualize the spatial correlation.

matlab

Copy code

% Plot Gaussian random field

figure;

surf(x, y, gaussian\_field\_grid);

title('Correlated Gaussian Random Field');

xlabel('X');

ylabel('Y');

zlabel('Amplitude');

shading interp;

colorbar;

% Plot Weibull random field

figure;

surf(x, y, weibull\_field\_grid);

title('Correlated Weibull Random Field');

xlabel('X');

ylabel('Y');

zlabel('Amplitude');

shading interp;

colorbar;

### **Complete MATLAB Code**

Below is the complete MATLAB code combining all the steps:

matlab

Copy code

% Define grid size

n = 50; % Number of grid points in each dimension

[x, y] = meshgrid(linspace(0, 1, n), linspace(0, 1, n));

grid\_points = [x(:), y(:)]; % Flatten the grid into a list of points

% Define correlation length

correlation\_length = 0.1;

% Number of points

num\_points = size(grid\_points, 1);

% Initialize covariance matrix

covariance\_matrix = zeros(num\_points);

% Compute pairwise distances and covariance

for i = 1:num\_points

for j = i:num\_points % Compute only upper triangle (symmetry)

distance = norm(grid\_points(i,:) - grid\_points(j,:));

covariance = exp(-distance / correlation\_length);

covariance\_matrix(i,j) = covariance;

covariance\_matrix(j,i) = covariance; % Symmetric matrix

end

end

% Cholesky decomposition of the covariance matrix

L = chol(covariance\_matrix + 1e-6 \* eye(num\_points), 'lower'); % Add small value to diagonal for numerical stability

% Generate uncorrelated standard normal random variables

z = randn(num\_points, 1);

% Generate correlated Gaussian random field

gaussian\_field = L \* z;

% Reshape to 2D grid for visualization

gaussian\_field\_grid = reshape(gaussian\_field, n, n);

% Weibull distribution parameters

shape\_param = 2.0; % k (shape parameter)

scale\_param = 1.0; % lambda (scale parameter)

% Convert Gaussian values to uniform values using the standard normal CDF

uniform\_values = normcdf(gaussian\_field);

% Apply the inverse Weibull CDF (quantile function)

weibull\_field = scale\_param \* (-log(1 - uniform\_values)).^(1 / shape\_param);

% Reshape to 2D grid for visualization

weibull\_field\_grid = reshape(weibull\_field, n, n);

% Plot Gaussian random field

figure;

surf(x, y, gaussian\_field\_grid);

title('Correlated Gaussian Random Field');

xlabel('X');

ylabel('Y');

zlabel('Amplitude');

shading interp;

colorbar;

% Plot Weibull random field

figure;

surf(x, y, weibull\_field\_grid);

title('Correlated Weibull Random Field');

xlabel('X');

ylabel('Y');

zlabel('Amplitude');

shading interp;

colorbar;

### **Explanation of Key Steps**

* **Covariance Matrix Computation**: The covariance between any two points decreases exponentially with distance, controlled by the correlation length. The covariance matrix is symmetric and positive-definite.
* **Cholesky Decomposition**: Decomposes the covariance matrix into a lower triangular matrix LL such that C=LL⊤C=LL⊤. Multiplying LL with a vector of uncorrelated standard normal random variables zz introduces the desired spatial correlation.
* **Transformation to Weibull Distribution**:
  + **Standard Normal CDF**: Converts the Gaussian random variables to uniform random variables in the range [0, 1].
  + **Inverse Weibull CDF**: Transforms uniform random variables to follow a Weibull distribution using the inverse CDF formula.

### **Considerations and Tips**

* **Numerical Stability**: When performing Cholesky decomposition, it's common to add a small value (e.g., 1e−61e−6) to the diagonal elements of the covariance matrix to ensure it is positive-definite and to improve numerical stability.
* **Correlation Length**: A smaller correlation length results in a rapidly decaying covariance function, leading to less spatial correlation between points. Adjust the correlation\_length parameter to control the extent of spatial correlation.
* **Grid Resolution**: Increasing the grid size n improves the spatial resolution but increases computational load, especially when computing the covariance matrix and performing Cholesky decomposition.
* **Alternative Methods**: For large grids, computing the full covariance matrix becomes computationally expensive. In such cases, consider using:
  + **Spectral Methods**: Utilize Fourier transforms to generate the random field more efficiently.
  + **Circulant Embedding**: Exploit the structure of the covariance matrix to reduce computational complexity.
  + **Gaussian Markov Random Fields (GMRF)**: Use sparse precision matrices instead of dense covariance matrices.

### **Summary**

By following the steps above, you can generate a spatially correlated random field that follows a Weibull distribution with a specified correlation length in MATLAB. This method combines spatial statistics and probability distribution transformations to create a field that captures both the desired spatial dependence and marginal distribution characteristics.

**Feel free to adjust the parameters (grid size, correlation length, Weibull parameters) to suit your specific application or to experiment with different spatial correlation structures.**