TECHNICAL UNIVERSITY OF CRETE

Boltzmann-Gibbs Local Interaction Models for Spatial Regression Problems

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Toganidis Nikolaos

Advisory Committe:

Professor Hristopulos T. Dionissios (supervisor), School of Electrical and Computer Engineering, Technical University of Crete

Professor Karystinos Georgios, School of Electrical and Computer Engineering, Technical University of Crete

Professor Spyropoulos Thrasyvoulos, School of Electrical and Computer Engineering, Technical University of Crete

July 22, 2024

A thesis submitted in fulfillment of the requirements for the diploma of Electrical and Computer Engineer

Abstract

Machine learning and Geostatistics are powerful mathematical frameworks for modeling spatial data. Both approaches, however, suffer from poor scaling of the required computational resources for large data applications. In 2015 the Stochastic Local Interaction (SLI) model was presented, which combined machine learning and geostatistics and with its local representation employment improved computational efficiency. Though, the model is very stable and performs well, its performance relies on some assumptions that not always hold. Specifically, due to it is based on a joint probability density function (Boltzmann-Gibbs pdf) defined by an energy functional and is expressed in terms of explicit, typically sparse, precision (inverse covariance) matrix, the curvature term (used to construct this precision matrix) must be semi-positive definite, and depends on the given dataset. In this thesis, in order, to eliminate those assumptions, the Graph Laplacian (GL) version of the model is introduced. Curvature terms now are calculated with the use of the second order of the Graph Laplacian, constructed using kernel functions and its local bandwidths to still keep the local representation employment and at the same time the low complexity, and prediction performance of the original model. This version, leads to a spatial analysis model, for any dimension with a good performance and low complexity, and now for any kind of spatial dataset.

Acknowledgements

Embarking on this academic journey has been one of the most significant challenges of my life, yet it has been immensely rewarding thanks to the support and guidance of many. At this moment of accomplishment, I am filled with gratitude for those who have been instrumental in my journey towards completing this thesis.

First and foremost, I would like to express my deepest gratitude to my thesis supervisor, Hristopulos Dionissios, whose expertise, consistency, understanding, and patience added considerably to my graduate experience. Your guidance and mentorship have been invaluable throughout this journey, not only in conducting the research but also in providing constant support and encouragement. I am eternally grateful.

I must also extend my heartfelt thanks to my friends, especially to Rania D., Konstantinos M., Nikolas P., Dimitris P., Dimitris B., Thodoris K., Konstantinos K., Dimitris G., Vasilis M., Giorgos I., Symeon S., Evi T., who paint the gray moments of this journey and gave me courage, love, and made me better as an engineer and as a person.

To my family, who have given me the love and support needed to pursue my studies, thank you for being my foundation. Your belief in me has been a constant source of motivation. Last but not least, I would like to thank my thesis committee member, Karystinos Georgios, Spyropoulos Thrasyvoulos, for their valuable input and scholarly insights, and the Technical University of Crete (TUC) for being a conductive academic environment, by not only providing me with the necessary knowledge that an engineer needs, but also making me want to change the world for its better.

This thesis stands as a milestone in my academic journey, and I am grateful for everyone who has been part of it. Thank you for making this experience memorable and enriching.

Toganidis Nikolaos
July, 2024

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's data-driven world, spatial data is pivotal across diverse domains such as environmental science, urban planning, and public health. This data, essential for geographical decision-making, often suffers from gaps due to various collection constraints, necessitating robust interpolation methods to generate complete datasets for effective analysis. The importance of computational efficiency in processing such voluminous and complex datasets cannot be overstated, as it ensures timely and actionable insights.

Models like the Stochastic Local Interaction (SLI) model address these challenges by optimizing the handling of spatial dependencies through local interactions and graph-based methods. The SLI model enhances computational performance without compromising accuracy, offering scalable and flexible solutions for managing spatial data. Its capability to accurately interpolate missing data through an understanding of spatial structure makes it indispensable in contemporary spatial data analysis.

Furthermore, continuous optimization efforts are directed at refining these models to minimize assumptions that could lead to errors or reduced performance. By improving algorithmic complexity and maintaining or enhancing accuracy, these advancements help pave the way for a more efficient and insightful future in spatial data utilization. That is what this thesis is about.

Chapter 2

Mathematical Preliminaries

2.1 Graphs

The spectral graph theory studies the properties of graphs via the eigenvalues and eigenvectors of their associated graph matrices: the adjacency matrix and the graph Laplacian and its variants (Williams (2006)).

2.1.1 Basic graph notations and Definitions

A **graph** is a mathematical structure used to model pairwise relations between objects. We consider simple graphs (no multiple edges or loops) as $\mathbf{G} = \{V, E\}$. Graphs are composed of two primary components:

- **Vertices (or nodes)**: These are the fundamental units of the graph. Each vertex represents an object or a point. The $V(G) = \{v_1, ..., v_n\}$ is called the vertex set with $n = |V|$;
- **Edges**: These are connections between pairs of vertices. An edge represents a relationship or link between the vertices it connects. The $E(G) = \{e_{ij}\}$ is called the edge set with $m = |E|$. If an edge e_{ij} connects vertices v_i and v_j, then those 2 nodes are adjacent or neighbors. One possible notation for adjacency is $v_i \sim v_j$. If there is an edge between every pair of vertices, then the graph is **complete**.

Graphs can be classified based on the nature of their vertices and edges:

- **Undirected Graph**: An edge between two vertices does not have a direction. The edge (u, v) is identical to the edge (v, u).
- **Directed Graph (or Digraph)**: An edge has a direction, going from one vertex to another. The edge (u, v) is not the same as (v, u).
- **Weighted Graph**: Edges have weights (or costs) associated with them, which could represent distances, costs, or any other metric.
- **Unweighted Graph**: Edges do not have any weights associated with them.

2.1.2 Subgraph of a Graph / Path/ Forest

H is a **subgraph** of G, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. A subgraph H is an induced subgraph of G if two vertices of $V(H)$ are adjacent if and only if they are adjacent in G. The complete subgraph of a graph is called **clique**.
A **path** of k vertices is a sequence of k distinct vertices such that consecutive vertices are adjacent.
A **cycle** is a connected subgraph where every vertex has exactly two neighbors. A graph containing no cycles is a **forest** and a connected forest is a **tree**.

2.1.3 Adjacency Matrix & Degree Matrix

- **Adjacency Matrix**: An adjacency matrix is a square matrix used to represent a graph. In a graph with n nodes, the adjacency matrix is an n x n matrix, where each element $A[i][j]$ represents the presence or absence of an edge between node i and node j. The matrix is defined by:

$$\mathbf{A} := \begin{cases} A_{ij} = 1 & \text{if there is an edge } e_{ij} \\ A_{ij} = 0 & \text{if there is no edge} \\ A_{ii} = 0 & \end{cases}$$

- **Degree Matrix**: In order to define the degree matrix, the definition of degree of a node must be given. More specifically, the number of neighbors of a node v is called the degree of v and is denoted by $d(v)$,

$$d(v_i) = \sum_{v_i \sim v_j} e_{ij}.$$

So, a degree matrix is a diagonal matrix that contains information about the degrees of nodes in a graph. In a graph with n nodes, the degree matrix is an n x n diagonal matrix, where each diagonal entry $D[i][i]$ represents the degree of node i (the number of edges incident to node i).

In an undirected graph, the degree of a node is equal to the number of edges connected to that node. Therefore, for an undirected graph, the degree matrix is a diagonal matrix where $D[i][i]$ contains the degree of node i.

In a directed graph, the degree of a node can be split into in-degree and out-degree. In-degree is the number of edges pointing toward the node, and out-degree is the number of edges leaving the node. In this case, the degree matrix is still a diagonal matrix, but $D[i][i]$ contains the sum of the in-degree and out-degree of node i.

2.1.4 Eigenvalues & Eigenvectors

A is a real-symmetric matrix: it has n real eigenvalues and its n real eigenvectors form an orthonormal basis. Let $\{\lambda_1, \ldots, \lambda_i, \ldots, \lambda_r\}$ be the set of *distinct* eigenvalues. The eigenspace S_i contains the eigenvectors associated with λ_i:

$$S_i = \{x \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \lambda_i \mathbf{x}\}.$$

For real-symmetric matrices, the algebraic multiplicity is equal to the geometric multiplicity, for all the eigenvalues. The dimension of S_i (geometric multiplicity) is equal to the multiplicity of λ_i. If $\lambda_i \neq \lambda_j$ then S_i and S_j are mutually orthogonal.

2.1.5 Real-valued functions on graphs

We consider real-valued functions on the set of the graph's vertices, $\mathbf{f} : \mathcal{V} \to \mathbb{R}$. Such a function assigns a real number to each graph node. The function $\mathbf{f}$ is a vector indexed by the graph's vertices, hence $\mathbf{f} \in \mathbb{R}^n$. Notation: $\mathbf{f} = (f(v_1), \ldots, f(v_n)) = (f(1), \ldots, f(n))$. The eigenvectors of the adjacency matrix, $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, can be viewed as ***eigenfunctions***.

2.2 Laplacian Matrix or Graph Laplacian

2.2.1 Definition

Let's let $e_i \in \{0,1\}^n$ be the standard basis vectors (1 in the i-th coordinate, 0's elsewhere). A *Laplacian* of an undirected graph $\mathbf{G} = (V, E)$,

$$\mathbf{L_G} = \sum_{(i,j)\in E} (e_i - e_j)(e_i - e_j)^\top.$$

Each term $(e_i - e_j)(e_i - e_j)^\top$ is an $|V| \times |V|$ matrix that has $+1$ in the (i,i) and (j,j) coordinate, -1 in the (i,j) and (j,i) coordinate and the rest of the entries are all 0 (Horaud (2015)). Now, we define the following notation:

The degree of i in $\mathbf{G}$ is denoted as $d(i)$. Let $\mathbf{D}$ be the diagonal matrix (degree matrix) defined by $\mathbf{D} = \text{diag}(d(i))$, where $\mathbf{D}(i,i) = d(i)$. The adjacency matrix of graph is denoted as $\mathbf{A}$. With this notation we can write $\mathbf{L_G} = \mathbf{D} - \mathbf{A}$.

2.2.2 Laplacian of Weighted Graph

Define $\mathbf{W} = (w(i,j)) \in \mathbb{R}^{n\times n}$ where $w(i,j) = 0$ if $(i,j) \notin \mathbf{E}$ and $\mathbf{D} = \text{diag}(d(i))$, where $d(i) = \sum_{(i,j)\in\mathbf{E}} w(i,j)$. Then $\mathbf{L_G} = \mathbf{D} - \mathbf{W}$ and this matrix is going to be denoted sometimes by $\mathbf{L_{G,W}}$.

2.2.3 Useful Mathematical Definitions

- **Positive (Semi)-Definiteness**: Matrix $A \in \mathbb{R}^{n\times n}$ is **positive semidefinite**, if $x^\top Ax \geq 0$ for all $x \in \mathbb{R}^n$. If A is positive semidefinite we write $A \succeq 0$.

  Here are some relevant properties:

  (i) $A \succeq 0$.

  (ii) $A = VV^\top$ for some matrix V.

  (iii) A has all non-negative eigenvalues.

  The Laplacian matrix or Graph Laplacian is positive semidefinite and so, $\mathbf{L_G} \succeq 0$ (Eduardo Pavez (2016)).

2.3 Kernel Functions & Adaptive Bandwidths

2.3.1 Adaptive Bandwidths

In order to better understand the adaptive bandwidths in SLI model, consider them as analogous to edge weights in a graph. In the context of the SLI model's graph-based representation, the adaptive bandwidths determine the strength and range of the interactions (or relationships) between nodes (data points). These bandwidths play a critical role in defining how strongly each pair of nodes influences each other, which is conceptually similar to how edge weights function in a graph. The adaptive bandwidths specify the strength of the interaction between nodes. A larger bandwidth implies a stronger or more influential interaction, while a smaller bandwidth indicates a weaker interaction. The bandwidths are adaptive, meaning they can vary from one pair of nodes to another, depending on the local characteristics of the data. This adaptability allows the model to capture local variations in spatial relationships effectively.

The SLI model uses kernel functions to implement these interactions. The bandwidths are parameters of these kernel functions, determining the shape and range of the interaction between nodes.

The interactions defined by these adaptive bandwidths contribute to the construction of the sparse precision (inverse covariance) matrix in the SLI model. This matrix encapsulates the conditional dependencies between nodes, where the bandwidths (as edge weights) influence the matrix's structure and sparsity. In spatial data analysis, these adaptive bandwidths (edge weights) are crucial for accurately capturing the spatial dependencies and heterogeneity present in the data, which is essential for reliable modeling and prediction.

2.3.2 Kernel Functions

Kernel functions are mathematical tools used to measure the similarity or correlation between points, often in the context of machine learning and statistical models. Depending on the specific characteristics of the spatial data and the analysis objectives different types of kernel functions are used. Common choices include Gaussian kernels, exponential kernels, and others, each with its own properties and suitability for different types of spatial interactions.

Theoretical Aspect

Kernel functions are based on the idea of transforming the input space into a higher-dimensional feature space, where the relationships between data points become more discernible. This transformation is often implicit, meaning the high-dimensional space is not explicitly computed, but its properties are inferred from the kernel function.

Mathematical Aspect

A kernel function $K(x, y)$, where x and y are n dimensional inputs ($n = 1, 2, \dots$), takes two inputs (usually vectors representing data points) and outputs a real number that quantifies the similarity or correlation between these inputs. For spatial data, x and y could represent the coordinates or attributes of two points, and the output is a measure of how closely related these points are.

Below is the table of the most basic kernel functions :

Kernel Type	**Formula**
Linear Kernel	$K(x, y) = x^\top y$
Polynomial Kernel	$K(x, y) = (x^\top y + c)^d$
Gaussian Kernel	$K(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$
Exponential Kernel	$K(x, y) = \exp\left(-\frac{\|x-y\|}{2\sigma^2}\right)$
Laplacian Kernel	$K(x, y) = \exp\left(-\frac{\|x-y\|}{\sigma}\right)$
Sigmoid Kernel	$K(x, y) = \tanh(\alpha x^\top y + c)$
Quadratic Kernel	$K(x, y) = (1 - \|x - y\|^2)^2$
Tricube Kernel	$K(x, y) = (1 - \|x - y\|^3)^3$ if $\|x - y\| < 1$, else 0
Triangular Kernel	$K(x, y) = (1 - \|x - y\|)$ if $\|x - y\| < 1$, else 0

TABLE 2.1: Common Kernel Functions

Kernel Averages

For any two-point function $\Phi(\cdot)$, a local-bandwidth extension of the Nadaraya-Watson kernel-weighted average is used over the network of sampling points:

$$(\Phi(\cdot))_{\mathbf{h}} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} K_{ij}\Phi(\cdot)}{\sum_{i=1}^{N}\sum_{j=1}^{N} K_{ij}},$$

where $\mathbf{h} = (h_1, h_2, \ldots, h_N)^T$ is the vector of local bandwidths. The function $\Phi(\cdot)$ represents the distance between two points $\|s_i - s_j\|$ or the difference $x_i - x_j$ of the field values, or any other function that depends on the locations or the values of the field. The kernel average is normalized so as to preserve unity, i.e., $\langle 1 \rangle_h = 1$ for all possible point configurations.

2.4 Precision Matrices

Before defining the precision matrix, it is important to understand the statistical concept of "precision". Specifically, **precision** is a measure of observational error or a description of random errors, serving as an indicator of statistical variability. The precision of a measurement system, related to both reproducibility and repeatability, refers to the degree to which repeated measurements under unchanged conditions yield the same results. It is worth noting that other summary statistics of statistical dispersion, also referred to as precision (or imprecision), include the reciprocal of the standard deviation, the standard deviation itself, and the relative standard deviation. Additionally, the standard error and the confidence interval (or its half-width, known as the margin of error) are also considered measures of precision.

The precision matrix is the inverse of the covariance matrix, which means that if $\mathbf{C}$ is the covariance matrix of a vector $\mathbf{X}$, then the precision matrix $\mathbf{J}$ is defined as $\mathbf{J} = \mathbf{C}^{-1}$. In statistical terms, while a covariance matrix captures the covariance (or correlation) between different variables, the precision matrix represents the conditional dependencies between them. Its elements indicate the strength and nature of the relationships between variables. A zero entry in a precision matrix suggests that the corresponding variables are conditionally independent given the other variables. In other words if two dimensions i and j of a multivariate normal are conditionally independent, then the ij and ji elements of the precision matrix are 0. This means that precision matrices tend

to be **sparse** when many of the dimensions are conditionally independent, which can lead to computational efficiencies when working with them. It also means that precision matrices are closely related to the idea of **partial correlation**.
The precision matrix of real-valued random variables, if it exists, is positive definite and symmetrical.

2.5 Stochastic Processes

A **stochastic process** is a collection of random variables indexed by a parameter, typically representing time or space. It is used to model systems or phenomena that evolve in a probabilistic manner over time or space (Williams (2006)).

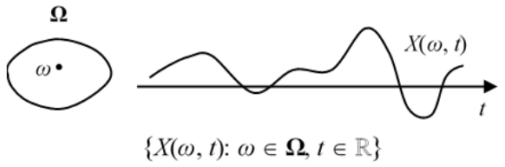

FIGURE 2.1: Example of a Stochastic Process

Definition: Formally, a stochastic process is defined as a family of random variables $\{X(\omega;t) : \omega \in \Omega, t \in \mathbb{R}\}$. Each random variable $X(t)$ is defined on a common probability space $(\Omega, \mathcal{F}, P)$. For a specific ω, $X(\omega;t)$ is causal function of t, while for a specific t, $X(\omega;t)$ is a random variable. $\{X(t), t \in \mathbb{R}\}$ can be denoted as a collection of random variables.

Every Stochastic Process is characterized from specific properties. The latter are mentioned bellow:

- **Stationarity**: A stochastic process is stationary if its statistical properties are invariant over time. Stationarity can be split in Wide-Sense (WSS) and Strict-Sense (SSS).
- **Ergodicity**: A stochastic process is ergodic if time averages converge to ensemble averages.
- **Independent Increments**: A stochastic process has independent increments if the increments over non-overlapping intervals are independent.

- **Markov Property**: A stochastic process has the Markov property if the future state depends only on the present state.

In addition, for a stochastic process $\{X(\omega;t) : \omega \in \Omega, t \in \mathbb{R}\}$, it is useful to define the most important functions:

i. **Mean Function:** The mean function $m_X(t)$ is defined as:

$$m_X(t) = \mathbb{E}[X(t)],$$

ii. **Covariance Function:**

The covariance function $C_X(t,s)$ is defined as:

$$C_X(t,s) = \mathbb{E}[(X(t) - m_X(t))(X(s) - m_X(s))],$$

iii. **Autocorrelation Function:**

The autocorrelation function $R_X(t,s)$ is defined as:

$$R_X(t,s) = \frac{C_X(t,s)}{\sigma_X(t)\,\sigma_X(s)}.$$

2.5.1 Gaussian Stochastic Processes

The class of Gaussian processes is one of the most widely used families of stochastic processes for modeling dependent data observed over time, or space, or time and space. The popularity of such processes stems primarily from two essential properties. First, a Gaussian process is completely determined by its mean and covariance functions. This property facilitates model fitting as only the first- and second-order moments of the process require specification. Second, solving the prediction problem is relatively straightforward. The best predictor of a Gaussian process at an unobserved location is a linear function of the observed values and, in many cases, these functions can be computed rather quickly using recursive formulas. The fundamental characterization, as described below, of a Gaussian process is that all the finite dimensional distributions have a multivariate normal (or Gaussian) distribution. In particular the distribution of each observation must be normally distributed. There are many applications, however,

where this assumption is not appropriate. For example, consider observations $x_1, \ldots, x_n$, where x_t denotes a 1 or 0, depending on whether or not the air pollution on the tth day at a certain site exceeds a government standard. A model for there data should only allow the values of 0 and 1 for each daily observation thereby precluding the normality assumption imposed by a Gaussian model. Nevertheless, Gaussian processes can still be used as building blocks to construct more complex models that are appropriate for non-Gaussian data.

Introduction

Gaussian stochastic processes are a powerful mathematical framework used to model a wide range of phenomena in fields such as statistics, machine learning, and geostatistics. In probability theory and statistics, a Gaussian process is a stochastic process such that every finite collection of those random variables has a multivariate normal distribution. The distribution of a Gaussian process is the joint distribution of all those (infinitely many) random variables, and as such, it is a distribution over functions with a continuous domain, e.g. time or space (D. Bertsekas, 2008). Before the definition of the Gaussian process, a few facts about Gaussian random vectors must be reminded. Random variables $X_1, X_2, \ldots, X_n$ are said to be **jointly normal** if, for all $a_1, a_2, \ldots, a_n \in \mathbb{R}$, the random variable

$$a_1 X_1 + a_2 X_2 + \ldots + a_n X_n$$

is a normal random variable. Also, a random vector

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$$

is said to be **normal** or **Gaussian** if the random variables $X_1, X_2, \ldots, X_n$ are jointly normal. An important property of jointly normal random variables is that their joint PDF is completely determined by their mean and covariance matrices. More specifically,

for a normal random vector $\mathbf{X}$ with mean $\mathbf{m}$ and covariance matrix $\mathbf{C}$, the PDF is given by

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{\det \mathbf{C}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m})\right\}.$$

Definition

A stochastic process $\{X(\omega;t) \mid \omega \in \Omega, t \in \mathbb{R}\}$ is called a **Gaussian Process**, if for every finite subset $\{t_1, t_2, \ldots, t_n\} \subset \mathbb{R}$, the random vector $(X(t_1), X(t_2), \ldots, X(t_n))$ has a multivariate normal (Gaussian) distribution.

Mathematically, a Gaussian process is defined by its mean function $m(t)$ and covariance function $k(t,t')$:

$$m(t) = \mathbb{E}[X(t)],$$

$$k(t,t') = \mathrm{Cov}(X(t), X(t')) = \mathbb{E}[(X(t)-m(t))(X(t')-m(t'))].$$

A Gaussian process can be written as:

$$X(t) \sim \mathcal{GP}(m(t), k(t,t')).$$

A Gaussian process has several important properties. Firstly, if $X(\omega;t)$ is a Gaussian process and a and b are constants, then the linear transformation $aX(t)+b$ results in another Gaussian process, demonstrating the property of **linearity**. Additionally, the **marginal** distribution of any subset of a Gaussian process remains Gaussian, highlighting the consistency of Gaussian distributions within the process. Moreover, the **conditional distribution** of a Gaussian process, given some observed values, is also a Gaussian process, maintaining the Gaussian nature even under conditioning on observed data.

Gaussian White Noise

A Gaussian white noise process $W(t)$ is a Gaussian process with mean function $m(t) = 0$ and covariance function $k(t,t') = \sigma^2\delta(t-t')$, where δ is the Dirac delta function. This implies that $W(t)$ has a constant power spectral density and that its values at different

times are uncorrelated. Mathematically, this means that for any two distinct times t and t', the random variables $W(t)$ and $W(t')$ are independent. Furthermore, each $W(t)$ follows a normal distribution with mean zero and variance σ^2. In practical applications, Gaussian white noise is often used to model random perturbations or disturbances in systems, providing a foundation for analyzing more complex stochastic processes.

Gaussian Process Regression

In Gaussian Process Regression (GPR), we aim to predict the value of a function at new points given some observed data points. Given a set of observed data points $\{(x_i, y_i)\}$, we assume:

$$y_i = f(x_i) + \epsilon_i \,,$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$ is Gaussian noise.

The joint distribution of the observed values and the function values at new points $\mathbf{X}_*$ is:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I & K(\mathbf{X}, \mathbf{X}_*) \\ K(\mathbf{X}_*, \mathbf{X}) & K(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right) ,$$

where $K(\mathbf{X}, \mathbf{X})$ is the covariance matrix evaluated at the training points, and $K(\mathbf{X}_*, \mathbf{X}_*)$ is the covariance matrix evaluated at the test points.

The posterior distribution is given by:

$$\mathbf{f}_* \mid \mathbf{X}, \mathbf{y}, \mathbf{X}_* \sim \mathcal{N}(\bar{\mathbf{f}_*}, \mathrm{Cov}(\mathbf{f}_*)) \,,$$

where

$$\bar{\mathbf{f}_*} = K(\mathbf{X}_*, \mathbf{X})[K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I]^{-1}\mathbf{y} \ ,$$

$$\mathrm{Cov}(\mathbf{f}_*) = K(\mathbf{X}_*, \mathbf{X}_*) - K(\mathbf{X}_*, \mathbf{X})[K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I]^{-1} K(\mathbf{X}, \mathbf{X}_*) \,.$$

In general, Gaussian processes have a wide range of applications across various fields due to their flexibility and powerful probabilistic framework. In machine learning, Gaussian processes are employed for regression tasks through Gaussian Process Regression (GPR), as well as for classification tasks, providing a non-parametric approach to learning. In geostatistics, Gaussian processes are utilized to model spatial data, allowing for

accurate predictions about unknown values based on observed data, which is essential for fields like environmental science and mining. Additionally, in time series analysis, Gaussian processes are applied to model and forecast time series data, capturing the underlying trends and uncertainties effectively. These applications highlight the versatility and importance of Gaussian processes in handling complex, real-world data.

Building on the foundational concepts of Gaussian stochastic processes, the discussion now extends to random fields, which generalize these processes to multi-dimensional spaces, providing a robust framework for modeling spatially and temporally varying phenomena.

2.6 Random Fields

A random field is a mathematical model that describes the spatial and/or temporal variations of a quantity that is random. Essentially, it is a generalization of a stochastic process where the underlying parameter space can be multi-dimensional (e.g., space and time). Formally, a random field is defined as a collection of random variables indexed by a multi-dimensional parameter, often representing spatial and/or temporal locations.

Definition: Let Ω be a probability space with a σ-algebra $\mathcal{F}$ and probability measure P. A random field $\{X(s) : s \in S\}$ is a collection of random variables $X(s)$ defined on Ω and indexed by a parameter s from a set S, which typically represents space and/or time.

The random fields are divided in 2 categories. More specifically, there are the **discrete random fields**, where the parameter space S is a discrete set, such as the integers or a finite grid and the **continuous random fields**, where the parameter space S is a continuous set, such as the real numbers or a continuous spatial domain.

Properties:

- **Mean Function**: The mean function of a random field $X(s)$ is defined as $m(s) = \mathbb{E}[X(s)]$.
- **Covariance Function**: The covariance function $C(s,t)$ of a random field $X(s)$ is defined as $C(s,t) = \mathbb{E}[(X(s) - m(s))(X(t) - m(t))]$.

- **Stationarity**: A random field is stationary if its statistical properties do not change when shifted in space and/or time.
- **Isotropy**: A random field is isotropic if its statistical properties are invariant under rotations.

2.6.1 Gaussian Random Fields

A Gaussian random field is a random field $\{X(s) : s \in S\}$ such that any finite subset $\{X(s_1), X(s_2), \ldots, X(s_n)\}$ follows a multivariate normal distribution. The mean function $m(s)$ and covariance function $C(s,t)$ completely characterize the field and the marginal distribution (percentages out of totals) at any location s is normal with mean $m(s)$ and variance $C(s,s)$ (Lindgren and Rue (2011)).
While Gaussian processes are typically used for time series data, Gaussian random fields are better suited for spatial data. Despite this difference, they share many common properties and underlying mathematical structures, making them versatile and widely applicable in various scientific and engineering disciplines (see subsection 2.5.1).

2.6.2 Markov Random Fields (MRF)

A Markov random field is a random field that satisfies the Markov property. The term Markov property refers to the memoryless property of a stochastic process, which means that its future evolution is independent of its history. A Markov random field extends this property to two or more dimensions or to random variables defined for an interconnected network of items. A discrete-time stochastic process satisfying the Markov property is known as a Markov chain (Rue and Held (2005)).

Definition: A random field $\{X(s) : s \in S\}$ is a Markov random field if for any $s \in S$, $X(s)$ is conditionally independent of all other random variables given its neighbors.

2.6.3 Spatial Random Fields (SRF)

A spatial random field is a type of random field where the index set represents spatial locations. Spatial random fields are used to model spatially varying phenomena where the observations at different locations are treated as random variables (Hristopulos (2020)).

Definition: A spatial random field $\{X(s,\omega) \in \mathbb{R}; s \in D(L) \subset \mathbb{R}^d; \omega \in \Omega\}$ is defined as a mapping from the probability space $(\Omega, \mathcal{A}, P)$ into the space of real numbers so that

for each fixed s, $X(s,\omega)$ is a measurable function of ω. $D(L)$ is the domain within which the SRF is defined and L is a characteristic domain length. An SRF involves many possible states, denoted by ω. In the following, for notational simplicity, we suppress the dependence on ω.

The realization of a particular state is determined from a joint probability density function (p.d.f.), $f_X[X(s)]$. The p.d.f. depends on the spatial configuration of the field's point values. For spatial data, the term sample refers to N values $X(s_i)$ from a particular state at the measurement locations $\{s_i, i = 1, \ldots, N\}$, representing a single state of the SRF (an observed realization).

For irregularly-spaced samples, practical applications involve determining the statistical parameters of spatial dependence and interpolating the data on a regular grid. An observed realization, $X^*(s)$, can be decomposed into a deterministic trend $m_X(s)$, a correlated fluctuation SRF $X_f(s)$, and a random noise term, $\epsilon(s)$, i.e., $X^*(s) = m_X(s) + X_f(s) + \epsilon(s)$. The trend is a non-stationary component representing large-scale, deterministic variations, which presumably correspond to the ensemble average of the SRF, i.e., $m_X(s) = \mathbb{E}[X(s)]$.

The fluctuation corresponds to variations that involve smaller spatial scales than the trend. It is assumed that resolvable fluctuations exceed the spatial resolution λ. The component $\epsilon(s)$ represents inherent variability below the resolution cutoff, or completely random variability due to uncorrelated measurement errors. It is assumed that $\epsilon(s)$ is statistically independent of the SRF $X_f(s)$, and can be treated as Gaussian white noise.

Furthermore, a stationary SRF is *statistically isotropic* if the covariance function depends only on the Euclidean distance between points but not on the direction of the lag vector, i.e., $G_X(r)$, where $\|r\|$ is the Euclidean norm of the vector r. The isotropic assumption is not restrictive, since the anisotropic parameters can be inferred from the data and isotropy can be restored by rotation and rescaling transformations.

For isotropic, short-ranged SRFs, one can define a single integral scale, given by the integral of the covariance function along any direction in space. Since the parameters of the fluctuation SRF are determined from the available sample by employing the *ergodic hypothesis*, the integral scale must be considerably smaller than the domain size L.

The distribution of *Gibbs random fields* is expressed in terms of an energy functional $H[X(s);\theta]$, where θ is a set of model parameters as follows:

$$f_X[X(s);\theta] = \frac{\exp\{-H[X(s);\theta]\}}{Z(\theta)}. \tag{2.1}$$

The constant $Z(\theta)$, called the partition function, normalizes the p.d.f. and is obtained by integrating $\exp\{-H[X(s);\theta]\}$ over all the realizations.

2.6.4 Gibbs Measure

A Gibbs measure is a probability measure used in statistical mechanics and probability theory to describe the distribution of states in a system with many interacting components. It generalizes the Boltzmann distribution to systems with more complex interactions and is fundamental in the study of random fields, particularly Markov random fields and spatial random fields.

2.6.5 Boltzmann-Gibbs Spatial Random Fields (G-SRF)

Boltzmann–Gibbs random fields are defined by means of exponential "joint-density" expressions, i.e.,

$$\rho[x(s)] = Z^{-1}\exp(-\mathcal{H}[x(D)]),$$

where Z is a normalization constant, known as the partition function and $x(D) = \{x(s), s \in D\}$ denotes a field configuration over the domain D. The calculation of Z is not straightforward, or even mathematically well-defined for infinite-dimensional configuration spaces (i.e., if the field states are continuous functions). Nonetheless, this definition of Boltzmann–Gibbs random fields is used in statistical field theory.

Definition: Let $\{X_s : s \in S\}$ be a collection of random variables indexed by a spatial parameter $s \in S$. The joint distribution of $\{X_s\}$ is a Gibbs measure if it can be written as:

$$P(X = x) = \frac{1}{Z}\, e^{-\beta H(x)},$$

where $H(x)$ is the Hamiltonian, β is the inverse temperature, and Z is the partition function (Hristopulos, 2022).

The pdf of *Gibbs SRFs* can be expressed in terms of an energy functional $H(x_S;\theta)$, where θ is a set of *model parameters*, according to the Gibbs pdf:

$$f_X(\mathbf{x_S};\boldsymbol{\theta}) = \frac{e^{-H(\mathbf{x_S};\boldsymbol{\theta})}}{Z(\boldsymbol{\theta})}. \tag{2.2}$$

Let $\mathbf{Q}(\theta)$ represent the following second-degree polynomial of the Laplace operator $\Delta = \nabla^2$,

$$\mathbf{Q}(\theta) = \theta_0 - \theta_1\Delta + \theta_2\Delta^2, \tag{2.3}$$

where $\theta = (\theta_0, \theta_1, \theta_2)^\top$ is a vector of polynomial coefficients and $\Delta^2 = (\nabla^2)^2$ is the Bi-Laplacian.

A Gaussian joint *probability density function (pdf)* can be defined as $\rho = Z^{-1}\exp(-\mathcal{H}[x(D)])$, with the following energy functional:

$$\mathcal{H}[x(D)] = \frac{1}{2}\langle x(s), Q\,x(s)\rangle, \tag{2.4}$$

Using integration by parts, the quadratic energy functional Eq. (2.4) can be expressed as follows:

$$\mathcal{H}[x(D);\theta] = \frac{1}{2}\left[\theta_0\langle x(s), x(s)\rangle + \theta_1\langle \nabla x(s), \nabla x(s)\rangle + \theta_2\langle \nabla^2 x(s), \nabla^2 x(s)\rangle\right]. \tag{2.5}$$

Chapter 3

Machine Learning

The integration of ML techniques with models such as the Stochastic Local Interaction (SLI) model enhances the capability to handle large-scale and high-dimensional spatial datasets. This chapter will explore how ML methodologies, particularly the K-st Nearest Neighbor, are used in the SLI model to optimize spatial data analysis, offering a substantial improvement over traditional statistical approaches.

3.1 Machine Learning and Geostatistics

Recent advancements in both machine learning and geostatistics have significantly impacted the analysis and prediction of spatial data in earth sciences. While geostatistics provides interpretable models for spatial and spatiotemporal dependence, machine learning offers powerful tools for identifying hidden structures in large datasets. The integration of these two fields has opened new avenues for more accurate and computationally efficient models of spatial processes.

Geostatistics and machine learning, despite their different origins, share many concepts and methodologies. For instance, kriging in geostatistics aligns closely with Gaussian process regression in machine learning. Both rely on the theory of Gaussian random fields and utilize positive definite functions (covariance functions in geostatistics and covariance kernels in machine learning) for interpolation and prediction tasks. This convergence allows for hybrid approaches that leverage the strengths of both fields to address complex spatial problems.

A critical challenge in both machine learning and geostatistics is handling large datasets, which often necessitates the inversion of large covariance matrices—a computationally

intensive task. Solutions to this include the Stochastic Partial Differential Equation (SPDE) approach, the Stochastic Local Interaction (SLI) model, and composite likelihood methods, all of which aim to improve computational efficiency by exploiting sparsity in data structures (Sandra De Iaco, 2022).

The integration of machine learning and geostatistics provides a powerful framework for advancing spatial data analysis and modeling. The hybrid approaches highlighted in this special issue demonstrate the potential for improved accuracy, computational efficiency, and robustness in addressing spatial and spatiotemporal challenges in the earth sciences.

3.2 K-Nearest Neighbor (KNN)

Definition

The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today. The k value in the k-NN algorithm defines how many neighbors will be checked to determine the classification of a specific query point. For example, if $k = 1$, the instance will be assigned to the same class as its single nearest neighbor. Defining k can be a balancing act as different values can lead to overfitting or underfitting. Lower values of k can have high variance, but low bias, and larger values of k may lead to high bias and lower variance. The choice of k will largely depend on the input data as data with more outliers or noise will likely perform better with higher values of k. Overall, it is recommended to have an odd number for k to avoid ties in classification, and cross-validation tactics can help you choose the optimal k for the given dataset. The algorithm in many cases is used to find the neighbors of given data points (very useful for spatial data analysis). For each point to be classified or predicted, k-NN identifies the k closest points (neighbors) from the training dataset. The value of k is a hyperparameter that needs to be chosen based on the dataset and the specific problem. The algorithm is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.

Statistical Setting

Let the pairs $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$ taking values in $\mathbb{R}^d \times \{1, 2\}$, where Y is the class label of X, so that $X|Y = r \sim P_r$ for $r = 1, 2$ (and probability distributions P_r). Given some norm $\|\cdot\|$ on $\mathbb{R}^d$ and a point $x \in \mathbb{R}^d$, let $(X_{(1)}, Y_{(1)}), \ldots, (X_{(n)}, Y_{(n)})$ be a reordering of the training data such that $\|X_{(1)} - x\| \leq \cdots \leq \|X_{(n)} - x\|$.

Classification

While the KNN algorithm can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

For classification problems, a class label is assigned on the basis of a majority vote—i.e. the label that is most frequently represented around a given data point is used. While this is technically considered "plurality voting", the term, "majority vote" is more commonly used in literature. The distinction between these terminologies is that "majority voting" technically requires a majority of greater than 50%, which primarily works when there are only two categories. When you have multiple classes—e.g. four categories, you don't necessarily need 50% of the vote to make a conclusion about a class; you could assign a class label with a vote of greater than 25%. In order to determine which data points are closest to a given query point, the distance between the query point and the other data points will need to be calculated. These distance metrics help to form decision boundaries, which partitions query points into different regions. You can see some of the most common distance metrics and their equations in the following table:

Distance Metric	**Equation**
Euclidean	$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
Manhattan	$\sum_{i=1}^{n} \lvert x_i - y_i \rvert$
Minkowski	$(\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p)^{\frac{1}{p}}$
Chebyshev	$\max_i \lvert x_i - y_i \rvert$
Hamming	$\sum_{i=1}^{n} \mathbf{1}(x_i \neq y_i)$

TABLE 3.1: Common Distance Metrics

It is important to note that in the case of the SLI model and the GL version of it (section 4), the KNN algorithm is not used to directly perform classification. Instead, it is performing a k-nearest neighbors search, which is a common step used in various machine learning tasks, including classification.

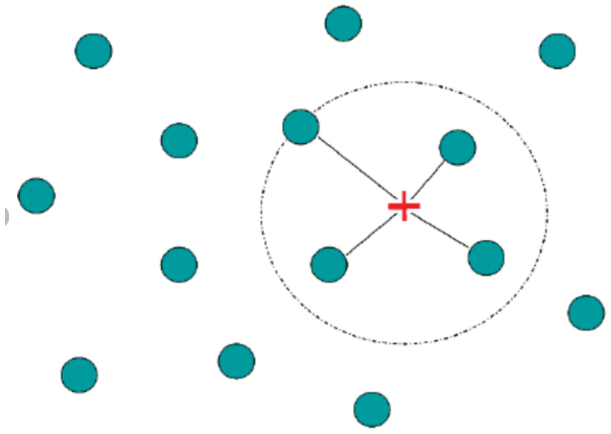

FIGURE 3.1: Example of KNN search for k=4

k-Nearest Neighbors (kNN) Search

k-Nearest Neighbors (kNN) search is a fundamental algorithm in data mining, pattern recognition, and machine learning, used to identify the closest data points (neighbors) to a given query point in a dataset. This search method is crucial in various applications, including classification, regression, and recommendation systems. The dataset consists of n data points, each represented as a vector in a d-dimensional space:

$$\{X_1, X_2, \ldots, X_n\}, \quad X_i \in \mathbb{R}^d$$

A query point $x \in \mathbb{R}^d$ is provided, and the goal is to find the k nearest neighbors from the dataset to this query point. The proximity of data points is typically measured using a distance metric. The default metric is usually Euclidean distance, but other metrics like Manhattan, Chebychev, Minkowski, and Mahalanobis distances can also be used. The Euclidean distance between two points X_i and X_j is calculated as:

$$\|X_i - X_j\| = \sqrt{\sum_{k=1}^{d} (X_{i,k} - X_{j,k})^2}$$

For each data point in the dataset, the distance to the query point x is computed. The data points are then sorted based on these distances, and the top k closest points are selected as the nearest neighbors.

3.3 Cross Validation

Definition

Cross Validation is a technique for assessing how the results of a statistical analysis will generalize to an independent data set. Cross-validation includes resampling and sample splitting methods that use different portions of the data to test and train a model on different iterations. It is often used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. It can also be used to assess the quality of a fitted model and the stability of its parameters

Leave-One-Out Cross Validation

Leave-one-out cross validation is K-fold cross validation taken to its logical extreme, with K equal to N, the number of data points in the set. That means that N separate times, the function approximator is trained on all the data except for one point and a prediction is made for that point. As before the average error is computed and used to evaluate the model. The evaluation given by leave-one-out cross validation error (LOO-XVE) is good, but at first pass it seems very expensive to compute. Fortunately, locally weighted learners can make LOO predictions just as easily as they make regular predictions. That means computing the LOO-XVE takes no more time than computing the residual error and it is a much better way to evaluate models. For example in SLI model the algorithmic complexity of SLI missing value estimation scales linearly with the sample size except for a global $\mathbf{O(N^2)}$ term which is, however, computed once for all the prediction points. Hence, the leave-one-out cross-validation approach can be used to efficiently infer the SLI model parameters.

Chapter 4

Methodological Background

4.1 Spatial data

Spatial data is any type of data that directly or indirectly references a specific geographical area or location. Sometimes called geospatial data or geographic information, spatial data most often numerically represents a physical object (natural or manmade) in a geographic coordinate system. The system uses two-dimensional (2D) geographic coordinates known as Cartesian or x and y coordinates. However, spatial data is much more than geographic coordinates represented on a 2D map. Spatial information can also include geometric shapes like lines or polygons, descriptions of a particular geographic feature, as well as images presented in the form of rasters (a grid of pixels).

We can categorize spatial data in 2 categories. More specifically, the 2 primary categories are the **geometric data**, which is mapped on a two-dimensional flat surface, such as a map or a floor plan (google maps), and **geographic** which is information mapped around a sphere, and highlights the latitude and longitude of a specific object or location.

In order to store spatial data, data formats are used. The most common ones are the vectors and rasters. The latter, also known as coverage data, is graphical data, usually presented in the form of bitmap images (although they can be stored in JPEG, TIFF, GIF and PNG formats as well). These images are compiled using pixels or tiny dots that form a pixel grid. Raster data is expressed as a range of values (over the pixelated grid). Each pixel stored within a raster has some value, usually about color or tone. The pixels

come together to create an image. The more pixels contained in an image and the more pixels displayed per inch, the higher its quality or sharpness. Resizing a raster image affects its quality since its pixels get stretched over a greater area.
From the other hand, vectors represent the real world in a graphical format. Vector data is most commonly used to represent features or objects on Earth's surface. These objects might be natural, such as trees, or manmade, such as roads or buildings and for this reason, vector data is also known as feature data.

4.2 Stohastic Local Interaction Model (SLI)

4.2.1 Overview

The Stochastic Local Interaction (SLI) model is a mathematical framework designed to handle, spatially correlated data which is based by construction on local correlations and is used for the interpolation of missing data through the dataset. Due to the focus on those local correlations of the data points, SLI has a low computational cost, a fact that is particularly useful in the context of big data (provides a computationally efficient method for modeling spatially correlated data, even in large-scale applications).
The local representation of spatial interactions, is achieved through the use of kernel functions with adaptive local bandwidths, allowing the model to handle local variations in data density and spatial relationships. More specifically, the SLI model is based on a joint probability density function (**Boltzmann-Gibbs PDF**) defined by an energy functional. This functional involves local interactions implemented by kernel functions, which are modeled in terms of a precision (inverse covariance) matrix, which is typically sparse (local relationships) and avoids the covariance matrix inversion (very high computational cost).
The predictive aspect of SLI involves maximizing the joint probability density of the data and the predictand, which is equivalent to minimizing the energy functional. This leads to a mode predictor with linear algorithmic complexity (Hristopulos (2015)).

4.2.2 Comparative Analysis of the SLI Model with Traditional Spatial Data Analysis Methods

Spatial analysis refers to studying entities by examining, assessing, evaluating, and modeling spatial data features such as locations, attributes, and relationships that reveal data's geometric or geographic properties. It uses a variety of computational models, analytical techniques, and algorithmic approaches to assimilate geographic information and define its suitability for a target system. Using the traditional spatial data analysis methods, it is easy to point out their limitations. Many of those models, have high computational costs (kriging, spatial clustering analysis), especially when it comes to large datasets, are limited in lower dimensions, or even have low accuracy when it comes to predictions (such as interpolation). In order to find solution to those limitations, the Stochastic Local Interaction model was developed, which due to its approach on the local interactions of the data points, and the use of a sparse precision matrix (constructed by those local correlations), it offers low computational cost, even for large datasets and is valid in any number of dimensions. Specialized, the algorithmic complexity scales **linearly** with the sample size except for a global $\mathbf{O}(\mathbf{N^2})$ term which is, however, computed once for all the prediction points, a complexity that allows the use of the leave-one-out cross-validation approach to efficiently infer the SLI model parameters.

Unlike traditional models that may falter with the complexity and scale of contemporary spatial datasets, SLI model enables efficient analysis of large-scale, high-dimensional data, which means that it can capture correlations in higher-dimensional, abstract feature spaces equipped with a suitable distance. Traditional approaches typically assume stationarity in spatial processes and apply fixed parameters, potentially overlooking local data variances. For example, while SLI is similar to the KNN (both methods employ an optimal neighborhood range), they differ in the way of the determination of the size of this neighborhood. In the case of KNN a uniform optimal number of nearest neighbors is determined, and the estimate at an unmeasured point is simply the mean of its k-nearest neighbors, the SLI however, a locally optimal neighborhood size is determined implying that the number of neighbors used in prediction varies locally and the estimate is a weighted mean of the neighbor values, in which the weights are determined by the kernel function and the bandwidths. SLI counters this by focusing on local interactions and employing adaptive bandwidths, thus finely tuning itself to local data density variations and spatial relationships for more accurate spatial dependency representation. In

addition, SLI can use weighted Euclidean distances or Minkowski metrics instead of the classical Euclidean distance and can also be extended to spherical surfaces, a case which is relevant for global geo-spatial data.

4.2.3 The Model

As mentioned in Subsection 3.2.1, the Stochastic Local Interaction (SLI) model is a mathematical framework designed to efficiently handle spatial data characterized by local correlations, offering low computational cost and effective interpolation for predicting missing data points within the dataset.

Energy Functional

The core of the SLI model is its energy functional, which determines the joint probability density function (pdf) of the model through a Gibbs distribution. The proposed energy functional is defined as follows:

$$H_X(\mathbf{x}_S;\boldsymbol{\theta}) = \frac{1}{2\lambda}\left[S_0(\mathbf{x}_S) + \alpha_1 S_1(\mathbf{x}_S;\mathbf{h}_1) + \alpha_2 S_2(\mathbf{x}_S;\mathbf{h}_2)\right], \tag{4.1}$$

where $\theta = (\mu_X, \alpha_1, \alpha_2, \lambda, \mu, k)$ represents the SLI parameter vector. The **Local Bandwidth Estimation Algorithm** is given below:

1. $h_{s,n} = \mu_s D_{n,[k]}(\mathcal{S}_N)$
2. $D_{n,[k]}(\mathcal{S}_N)$ is the distance between $\mathbf{s}_n$ and its k-nearest neighbor in $\mathcal{S}_N$
3. $\mu_s > 1$ is a **global** bandwidth parameter inferred from the data
4. Pre-define $k \in \{2, 3, 4, 5\}$ for **compactly supported** kernels

Same idea in time: $h_{t,n} = \mu_t(k_t - 1)\delta t$, where δt is the time step.

Square Fluctuations, Gradient & Curvature

The terms $S_0(\mathbf{x_S})$, $S_1(\mathbf{x_S};\mathbf{h_1})$, and $S_2(\mathbf{x_S};\mathbf{h_2})$ correspond to the averages of the square fluctuations, the square gradient, and the square curvature, respectively, in a Euclidean

space of dimension d. These terms are computed using kernel-weighted averages that involve the field increments $x_{ij} = x_i - x_j$. The formulas for these terms are:

$$S_0(\mathbf{x_S}) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)^2,$$

$$S_1(\mathbf{x_S}; \mathbf{h_1}) = \left\langle \left(\frac{x_i - x_j}{h_1} \right)^2 \right\rangle_{h_1},$$

$$S_2(\mathbf{x_S}; \mathbf{h_2}) = \left\langle \left(\frac{x_i - 2x_j + x_k}{h_2^2} \right)^2 \right\rangle_{h_2}.$$

Here, $\mathbf{h_1}$ and $\mathbf{h_2}$ are two vector bandwidths that determine the range of influence of the kernel function around each sampling point for the gradient and curvature terms, respectively.

To obtain realistic kernel bandwidths, the parameter k should be a positive integer greater than one ($k > 1$), and μ should be larger than one ($\mu > 1$).

Precision Matrix Representation

With the energy functional defined as above, it is straightforward to derive a precision matrix, $\mathbf{J}(\boldsymbol{\theta})$, which is explicitly defined in terms of local interactions and thus avoids the computationally expensive inversion of the covariance matrix. The energy functional can be expressed in terms of the precision matrix as follows:

$$H_X(\mathbf{x_S}; \boldsymbol{\theta}) = \frac{1}{2}(\mathbf{x_S} - \boldsymbol{\mu_X})^T \mathbf{J}(\boldsymbol{\theta})(\mathbf{x_S} - \boldsymbol{\mu_X}). \tag{4.2}$$

By expanding the squared differences in the energy functional, the following expression for the precision matrix is obtained:

$$\mathbf{J}(\boldsymbol{\theta}) = \frac{1}{\lambda}\left\{ \frac{\mathbf{I_N}}{N} + \alpha_1 d\mathbf{J_1}(\mathbf{h}_1) + \alpha_2 \left[c_{2,1}\mathbf{J_2}(\mathbf{h}_2) - c_{2,2}\mathbf{J_3}(\mathbf{h}_3) - c_{2,3}\mathbf{J_4}(\mathbf{h}_4) \right] \right\}, \tag{4.3}$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix: $[\mathbf{I}_N] = 1$, if $i = j$ and $[\mathbf{I}_N] = 0$ otherwise, and $\mathbf{J}_q(\mathbf{h}_q), q = 1, 2, 3, 4$ are network matrices that are determined by the sampling pattern, the kernel function, and the bandwidths. The index q defines the gradient network matrix for $q = 1$, whereas the values $q = 2, 3, 4$ specify the curvature network matrices that correspond to the three terms in $S_2(\mathbf{x_S}; \mathbf{h_2})$.

Network Matrices

The network matrices of the model are symmetric by construction, while the rows and columns sums vanish, i.e., $\sum_{j=1}^{N} [\mathbf{J}_q(\mathbf{h}_q)]_{ij} = 0$. However, in order to better understand those properties, the elements of network matrices must be defined. The latter, are given by the following equations:

$$\begin{aligned}[\mathbf{J}_q(\mathbf{h}_q)]_{ij} = &-u_{i,j}(h_{q,i}) - u_{i,j}(h_{q,j}) \\ &+ [\mathbf{I}_N]_{ij} \sum_{l=1}^{N} \left[u_{i,l}(h_{q,i}) + u_{l,i}(h_{q,l})\right], \end{aligned} \tag{4.4}$$

$$u_{i,j}(h_{q,i}) = \frac{K\left(\frac{s_i - s_j}{h_{q,i}}\right)}{\sum_{i=1}^{N}\sum_{j=1}^{N} K\left(\frac{s_i - s_j}{h_{q,i}}\right)}, \quad q = 1, \ldots, 4. \tag{4.5}$$

It is worth to note that network matrices or the sub-matrices $\mathbf{J}_q(\mathbf{h}_q)$ are **diagonally dominant**, since the kernel weights are non-negative.

Spatial Prediction

Having defined now the energy functional (in both forms) and the network matrices, it is easy to describe the predictive SLI model. Specifically, assume that the prediction point $\mathbf{s_p}$ is added to the sampling points. By inserting this point in the energy functional described in (3.2), and by defining the mode of the joint pdf, the unknown value of the field in $\mathbf{s_p}$ can be predicted. The prediction value is $\hat{\mathbf{x}}_p$ and the mode of the prediction equation for this value is given by the following equation:

$$\hat{x}_p = \mu_X - \frac{\sum_{i=1}^{N} \left[J_{i,p}(\boldsymbol{\theta}^*) + J_{p,i}(\boldsymbol{\theta}^*)\right](x_i - \mu_X)}{2J_{p,p}(\boldsymbol{\theta}^*)} = \mu_X - \frac{\sum_{i=1}^{N} J_{p,i}(\boldsymbol{\theta}^*)(x_i - \mu_X)}{J_{p,p}(\boldsymbol{\theta}^*)}\,. \quad (4.6)$$

It is easy to see, that the mode predictor of SLI can be generalized to P prediction points by using a $N \times P$ precision matrix $[\mathbf{J}_{P,S}(\boldsymbol{\theta}^*)]_{p,i} = \frac{J_{p,i}(\boldsymbol{\theta}^*)}{J_{p,p}(\boldsymbol{\theta}^*)}, \quad p = 1, \ldots, P; \quad i = 1, \ldots, N$. The SLI mode predictor for P points is given below:

$$\hat{\mathbf{x}}_p = \boldsymbol{\mu}_X - \mathbf{J}_{P,S}(\boldsymbol{\theta}^*)(\mathbf{x} - \boldsymbol{\mu}_X)\,. \quad (4.7)$$

While the precision matrix elements using the formulation of (3.4), the kernel weights are different, and thus must be defined. The updated kernel weights are a result of the inclusion of the $\mathbf{s_p}$ and can be easily defined with a simple modification of the (3.5).
Updated Kernel Weights :

$$u_{i,j}(h_{q,i}) = \frac{K\left(\frac{\mathbf{s_i}-\mathbf{s_j}}{h_{q,i}}\right)}{\sum_{i,j} K\left(\frac{\mathbf{s_i}-\mathbf{s_j}}{h_{q,i}}\right) + \sum_i K\left(\frac{\mathbf{s_i}-\mathbf{s_p}}{h_{q,i}}\right) + \sum_i K\left(\frac{\mathbf{s_i}-\mathbf{s_p}}{h_{q,p}}\right)}, \quad (4.8)$$

where the first term in the denominator concerns interactions between sampling points, the second term involves local interactions between the sampling points and the prediction point which result from inserting the prediction point in the local neighborhoods of the sampling points,which control the bandwidths. Finally, the third term also involves interactions between the prediction point and the sampling points. The index q distinguishes the weights linked to the gradient ($q = 1$) and to the curvature ($q = 2, 3, 4$).
Additionally, the weights that correspond to combinations of sampling and prediction points are defined as:

$$u_{p,j}(h_{q,p}) = \frac{K\left(\frac{\mathbf{s_p}-\mathbf{s_j}}{h_{q,p}}\right)}{\sum_{i,j} K\left(\frac{\mathbf{s_i}-\mathbf{s_j}}{h_{q,i}}\right) + \sum_i K\left(\frac{\mathbf{s_i}-\mathbf{s_p}}{h_{q,i}}\right) + \sum_i K\left(\frac{\mathbf{s_i}-\mathbf{s_p}}{h_{q,p}}\right)}, \quad (4.9)$$

where $p = 1, \ldots, P$ and $j = 1, \ldots, N$.

Lastly, with the definitions of the kernel weights after the inclusion of the $\mathbf{s_p}$, the elements of the precision matrix, that involve the prediction point can also be defined as follows:

$$[\mathbf{J}_q(\mathbf{h}_q)]_{p,p} = \sum_{i=1}^{N} [u_{i,p}(h_{q,i}) + u_{p,i}(h_{q,p})] , \tag{4.10}$$

$$[\mathbf{J}_q(\mathbf{h}_q)]_{i,p} = - [u_{i,p}(h_{q,i}) + u_{p,i}(h_{q,p})] , \quad i \neq p. \tag{4.11}$$

Observing (4.10), it can be easily seen that the symmetry property, $J_{p,i}(\boldsymbol{\theta}^*) = J_{i,p}(\boldsymbol{\theta}^*)$, follows. Furthermore, due to the different bandwidths used, the coefficients $u_{i,p}(h_{q,i})$ and $u_{p,i}(h_{q,p})$, differ (in the former, the bandwidth is determined by the neighborhood of the sampling point s_i, whereas in the latter by the neighborhood of s_p) (Dionissios T. Hristopulos (2020)).

Predictor Properties

Due to the fulfillment of the vanishing row sum property, $\sum_{j=1}^{N}[J(\boldsymbol{\theta})]_{i,j} = \frac{1}{N\lambda}$, imposed by the network matrices and the precision matrix, the SLI predictor (3.7) exhibits unbiasedness. In addition, the SLI predictor is independent of the parameter λ (amplitude of the fluctuations), because the transfer matrix $J_{p,s}(\boldsymbol{\theta}^*)$, is determined by the ratio of precision matrix elements, a property similar to the one of the kriging predictor from the random field variance. Thus, leave-one-out cross-validation does not influence the optimal value of λ.

It should be pointed out that the SLI predictor is not always an exact interpolator. Consider a point s_k where $k \in \{1, \ldots, N\}$, which is very close to s_p. Based on (4.10), (4.11) and (4.7), $\hat{x}_p \to x_k$ as $s_p \to s_k$ only if (i) $u_{k,p}(h_p) \gg u_{i,p}(h_p)$ and (ii) $u_{k,p}(h_p) \gg u_{i,p}(h_i)$ for all $i \neq k$.

Conditions:

i. Only for compactly supported kernels if $h_p \to 0$, which necessitates that the bandwidth be determined by the nearest neighbor distance.

ii. Requires that $\|s_k - s_p\|/h_k \ll \|s_i - s_p\|/h_i$ for $i \neq k$. This condition is approximately satisfied at best if the sample is sparse around s_p.

It is crucial to emphasize that the model is specifically designed for the **interpolation** of data. The underlying reason it is not suitable for extrapolation is that it relies on local neighborhood information, particularly on data points that are locally correlated. So, due to Extrapolation is the process of ascertaining missing data points within an "unknown" data range (in contrast interpolation is used to ascertain missing data points within a "known" data range), the distances between data points can become excessively large, which can lead to significant inaccuracies and potential issues in the model's predictions (Michael McCartney (2020)).

Parameter Inference

Experiments with both maximum likelihood estimation and leave-one-out cross validation were done. The former requires the calculation of the SLI partition function, which is an $O(N^3)$ operation for scattered data. For large data sets the $O(N^3)$ complexity is a computational bottleneck. Parameter inference by optimization of a cross validation metric is computationally more efficient, since it is at worst an $O(N^2)$ operation The memory requirements for storing the precision matrix are $O(N^2)$ but can be significantly reduced by using sparse matrix structures. Let $\boldsymbol{\theta}_{-\lambda} = (\alpha_1, \alpha_2, \mu, \mu_X)^T$ represent the parameter vector excluding λ. The following cross validation cost functional was used:

$$\Phi(\mathbf{x_S}; \boldsymbol{\theta}_{-\lambda}) = \sum_{i=1}^{N} |\hat{x}_i(\boldsymbol{\theta}_{-\lambda}) - x_i| \,, \tag{4.12}$$

where $\hat{x}_i(\boldsymbol{\theta}_{-\lambda})$ is the SLI prediction at s_i based on the reduced sampling set $S_N - \{s_i\}$ using the parameter vector $\boldsymbol{\theta}_{-\lambda}$, which applies to all $i = 1, \ldots, N$. The prediction is based on the interpolation equation and does not involve λ.

The **optimal parameter vector** excluding λ, i.e., $\theta_{-\lambda}$, is determined by minimizing the cost functional (14):

$$\boldsymbol{\theta}^*_{-\lambda} = \arg\min_{\boldsymbol{\theta}_{-\lambda}} \Phi(\mathbf{x_S}; \boldsymbol{\theta}_{-\lambda}). \tag{4.13}$$

If $H(\mathbf{x_S}; \boldsymbol{\theta}_{-\lambda})$ is the energy estimated from (4.1) and (4.2) by setting $\lambda = 1$, the optimal value λ^* is obtained by minimizing the negative log-likelihood with respect to λ leading to the following solution (Hristopulos (2015)):

$$\lambda^* = \frac{2H(\mathbf{x_S}; \boldsymbol{\theta}_{-\lambda})}{N}. \tag{4.14}$$

The determination of the minimum of the cross validation cost functional (4.12) happened using the MATLAB constrained optimization function `fmincon` with the interior-point algorithm. This function determines the local optimum nearest to the initial parameter vector. We use initial guesses for the parameters α_1, α_2, μ, and we assume that the parameters are constrained between the lower bounds $[0.5, 0.5, 0.5]$ and the upper bounds $[300, 300, 15]$. We investigated different initial guesses for the parameters which led to different local optima. We found, however, that the value of the cross validation function is not very sensitive on the local optimum.

4.3 SLI Model - Graph Laplacian Edition

As described previously, SLI model must satisfy certain conditions in order to be permissible. Specifically, its energy functional, $H_x(\mathbf{x_s}; \theta)$, is permissible if $H_x(\mathbf{x_s}; \theta) \geq 0$ for all the $\mathbf{x_s}$, a condition that ensures that boundaries for the term $e^{-H(\mathbf{x}_s;\boldsymbol{\theta})}$, exist, and also the existence of the partition function, $Z(\boldsymbol{\theta})$, in the Gibbs pdf, $f_X(\mathbf{x}_s; \boldsymbol{\theta}) = \frac{e^{-H(\mathbf{x}_s;\boldsymbol{\theta})}}{Z(\boldsymbol{\theta})}$. However, the energy functional is constructed with the usage of the S_2 term for the calculation of the curvature, either from the (4.1), or with the precision matrix implementation (4.2). The problem is that for the curvature to satisfy the above condition, an assumption must be made: $S_2 \geq 0$ (Note: S_0 and S_1 are always non-negative by construction), and this is not guaranteed for all kinds of data. Thus, to make sure that the curvature is always going to be non-negative, the S_2 term is replaced by the second order of the graph Laplacian (the graph Laplacian by its definition is always positive semi-definite).

Implementation of Graph Laplacian

In this solution, the energy functional used, is in the (4.2) form, and so the second order graph Laplacian must be a matrix with the same dimensions as the one in the previous solution, for S_2 (since the rest terms are not changed). In order to calculate the graph Laplacian, the formula, $\mathbf{L} = \mathbf{D} - \mathbf{A}$, is used, where $\mathbf{L}$ is the graph Laplacian or Laplacian Matrix, $\mathbf{D}$ is the degree matrix and $\mathbf{A}$ is the adjacency matrix.
Let's start from the calculation of the adjacency matrix, which considers as neighbors only the points that are locally correlated. Firstly, the distance matrix is calculated the same way as in the previous implementation, by using nearest neighbors for infinitely supported kernels and second nearest neighbors for compactly supported kernels. Having, the distance matrix which contains the distances among all the points, it is easy now to

calculate the adjacency matrix. It is important to check before the calculation, the kernel function that is used, because the Gaussian, the Exponential, due to their formula, may give values that tend to infinity and so a threshold must be made to bound those values (in this implementation threshold is 0.8 which is chosen after the experiments). After the above, the distance matrix is given as an input to the kernel function that is used, which is going to result a new sparse (in most cases) matrix containing the distances of the local neighbors only. Lastly, the diagonal matrix must be subtracted in order to gain the Adjacency one. Note, that this method is for the sample to sample points.

Having the adjacency matrix it only remains to calculate the degree matrix and then use the formula above for the Laplacian. The degree matrix is calculated through the adjacency by taking the sum of the edges leaving or inserting the node i, $\mathbf{D} = \text{diag}(d_1, \ldots, d_n)$. Now, the Laplacian matrix or the Graph Laplacian can easily be calculated through the formula: $\mathbf{L} = \mathbf{D} - \mathbf{A}$. However, a focus must be made for the calculation of its second order. More specifically, the Laplacian Matrix must be normalized by the second order of the kernel bandwidth. This happens because it adjusts the scale of the spatial interactions to be consistent across different areas of the dataset, especially when there are variations in data density or when spatial relationships vary across the dataset. It ensures that the influence or weight assigned by the kernel function remains consistent and meaningful relative to the spatial scale of the data and helps maintain numerical stability and improve the accuracy of the computations. Lastly, normalizing by the bandwidth makes the Laplacian matrix entries more interpretable, as they directly reflect the relative strength of connections after accounting for local variations in point density and distribution. Having the normalized Laplacian it is easy to calculate the precision matrix of the energy functional, by replacing the precision matrix of the previous model representing the S_2 term with the normalized Laplacian matrix.

Adaptive Bandwidths

In the case of the original SLI model the bandwidths for gradient ($\mathbf{h_1}$) and curvature ($\mathbf{h_2}$) are chosen through K-Nearest Neighbor search. These bandwidths are used in order to determine the shape and range of the interaction between nodes. The ratios of the gradient and curvature bandwidths are determined based on the values of the third nearest neighbors of the sampling points (nearest neighbors are at zero distance). The

same way as described above is used to determine the bandwidths for the case of the SLI-GL.

More specifically, thhe local bandwidth associated with s_i is chose according to

$$h_i = \mu D_{i,k}(S_N), \tag{4.15}$$

where $\mu > 1$ and $k > 1$ are model parameters. In several case studies involving Euclidean spaces of dimension $d = 1, 2, 3, 4$, is determined that $k = 2$ (second nearest neighbors) performs well for compactly supported kernels and $k = 1$ (nearest neighbors) for infinitely supported kernels. Using $k = 2$ for compact kernels avoids zero bandwidth problems which result from $k = 1$ for collocated sampling and prediction points. Since the sampling point configuration is fixed, μ and $D_{i,k}(S_N)$ determine the local bandwidths. $D_{i,k}(S_N)$ depends purely on the sampling point configuration, but μ also depends on the sample values. For compactly supported kernels setting $k = 1$ only makes sense if $\mu > 1$; otherwise

$$h_i = \mu D_{i,k=1}(S_N) \tag{4.16}$$

implying that the kernel vanishes even for the nearest-neighbor pairs and thus fails to implement interactions.

Cases where the dimensions are more than 3 must be handled with caution, specially when the data points are not enough. Specifically, it may need to determine the bandwidths according to the second nearest neighbor, cause in other case it is possible to cover the whole area due to the sparseness of the data points in the space.

Graph Laplacian between Prediction and Sample Points

The above methodology is for the calculation of the matrix between sample to sample points. While the methodologies for calculating both precision matrices (sample to sample and predicted to sample) are similar, there are some key differences that must be noted. First of all, the predicted values, P, most of the cases are not equal to the sample ones, N, $N \neq P$, which will result a non square matrix. This is causing a problem for the calculation of the Laplacian due to the definition of both adjacency and Laplacian that must be square matrices. So, in order to fix this problem, a global matrix containing all the points is created, with dimensions of $(P + N) \times (P + N)$. So having this global square matrix it is feasible to calculate from it its adjacency (after the distance matrix)

and then with the same methodology to gain the Laplacian one. Having the Laplacian global matrix, it only remains to gain the first P rows and the last N columns to have the Laplacian matrix of the predicted to sample. The methodology after that is the same with the one described above.

Chapter 5

Design of Numerical Experiments and Analysis of Results

It is time to prove and verify that the new model created to eliminate the assumptions discussed previously, the implementation with the second order of the graph Laplacian, has the same (and better) prediction accuracy, without increasing the complexity. Firstly, there will by a verification of the method created for calculating the graph Laplacian, through a toy example and then, the 2 models will be compared through 2 experiments, one for 2 dimensions and one for 4. Also a comparative analysis will be provided, showing the error metrics.

5.1 Validation of Graph Laplacian Implementation

In order to verify that the method for calculating the graph Laplacian is valid, a comparison on a grid with values for both x and y from -10 to 10 with a spacing of 0.5, and values the result of the eggholder function was made. The eggholder function can be seen next:

$$f(x,y) = -(y+47)\sin\left(\sqrt{\left|y+\frac{x}{2}+47\right|}\right) - x\sin\left(\sqrt{|x-(y+47)|}\right) . \tag{5.1}$$

The Laplacian method of the SLI, was compared with the result of MATLAB's, `del2` function applied to the values of (5.1) and their difference was visualized in a heatmap. The visualization can be seen below:

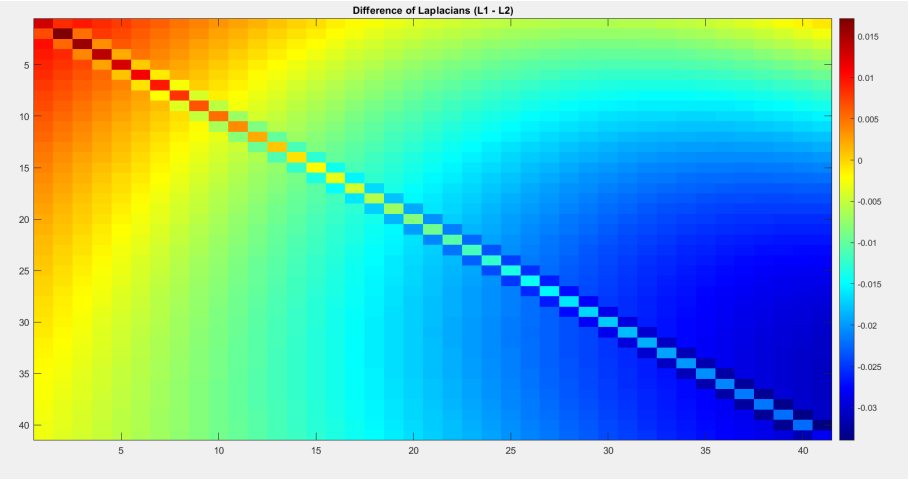

FIGURE 5.1: Heatmap of the difference between the Laplacian matrix of MATLAB's `del2` and the Laplacian matrix using the function of GL SLI model

While there are small systematic differences between the two methods, these differences are likely due to the aforementioned factors. Both methods can be considered valid cause they adhere to the same theoretical framework and produce similar results within acceptable numerical precision limits.

5.2 Analysis of Results

As mentioned before the 2 models (SLI, and SLI with graph Laplacian) need to be compared, to verify the performance of the new one. Through the following experiments, the performance of the new model will be verified (through comparison with the original SLI), by checking the precision matrix used in energy functional, the SLI sample-predictions, the smoothness of the map of the interpolated values, and the LVO cross validation SLI parameter estimates of SLI energy versus the respective removed sample value. All of the above have been visualized for better understanding.

5.2.1 Analysis of Radioactivity data

In order to get a better idea of the dataset of the experiment, the training set points and the 2D convex hall (domain boundary) are provided. The specific dataset is about Gamma dose rate radiation data over Germany (in nanosievert per hour). In addition, a colour map of Linear Interpolation of training set points and convex hull of the areas is provided, with the use of the MATLAB's functions, `griddata()` for the interpolation and `meshgrid()` for the creation of the grid, with the coordinates of the training set. Both the plots can be seen below:

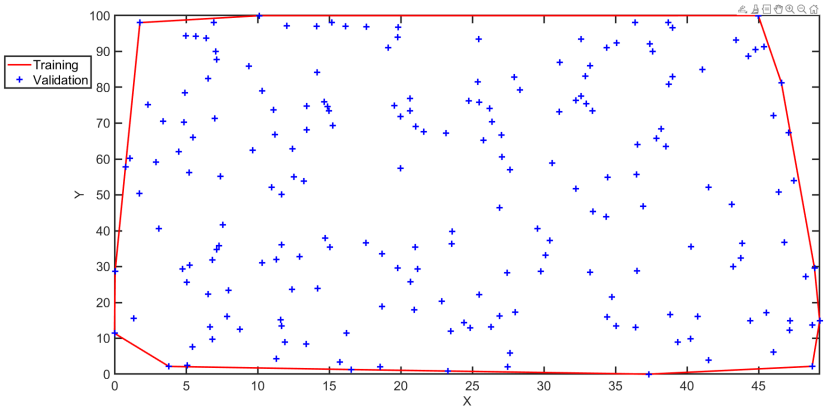

FIGURE 5.2: Training set points and Convex Hull (domain boundary)

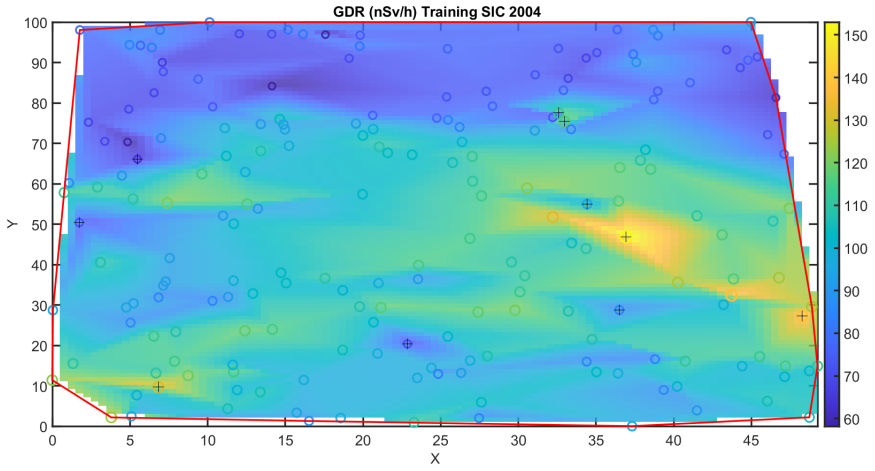

FIGURE 5.3: Linear Interpolation of training set point & Convex Hull of the areas

As described above both models are built with the implementation that uses a precision matrix. It is observed that in the implementation with the second order of the graph Laplacian, the precision matrix is more sparse, while in the original model it can be seen that high values are in many cells of the matrix. The precision matrix of the original SLI model has $10,786$ non zero elements, out of $40,000$ resulting a sparsity score of 73.035%,while the precision matrix of the GL version of SLI model has $2,608$ non zero elements, out of $40,000$ resulting a sparsity score of 93.800%. The visualizations of the logarithm of absolute values of the 2 precision matrices from the 2 models that represent the sample to sample points can be seen below:

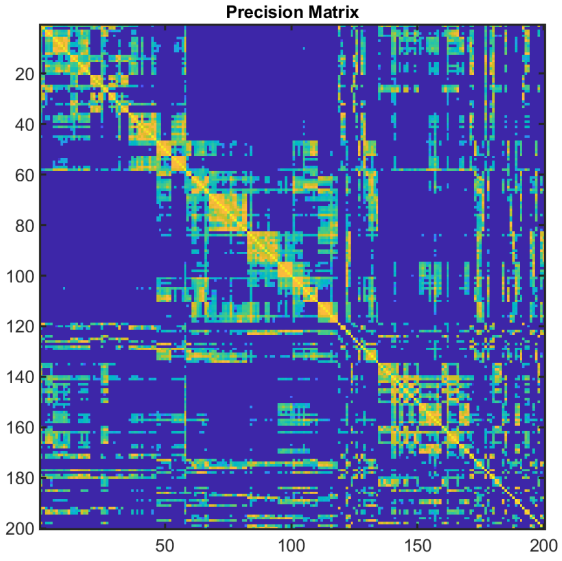

FIGURE 5.4: Logarithm of Absolute Value of Precision Matrix for sample to sample data points from the original SLI model

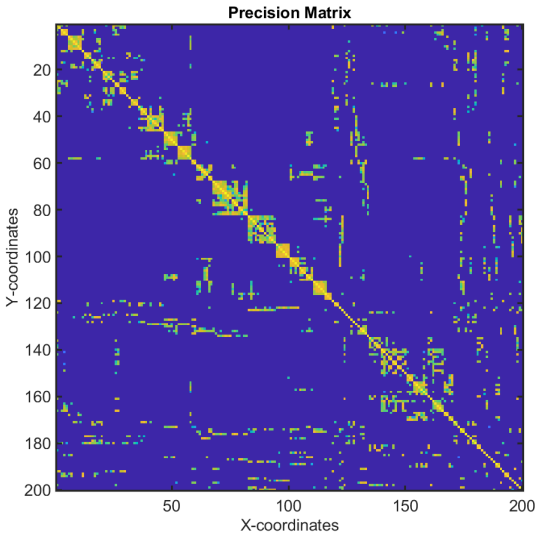

FIGURE 5.5: Logarithm of Absolute Value of Precision Matrix for sample to sample data points from the SLI model with graph Laplacian Implementation

It is easy to see now that the precision matrix of the new implementation is more sparse than the one of the original model, a fact that is going to reduce the SLI model's complexity.

As mentioned in the previous section, the way the precision matrix is calculated, it differs according the need. It is easier for sample to sample and more difficult for the sample to prediction points. So, in order to have a first picture of the behaviour of the

model and the interpolation accuracy, a validation was made through a leave one out cross validation for the sample to sample precision matrix. The results for both models (original and Graph Laplacian implementation) are plotted and can be seen next:

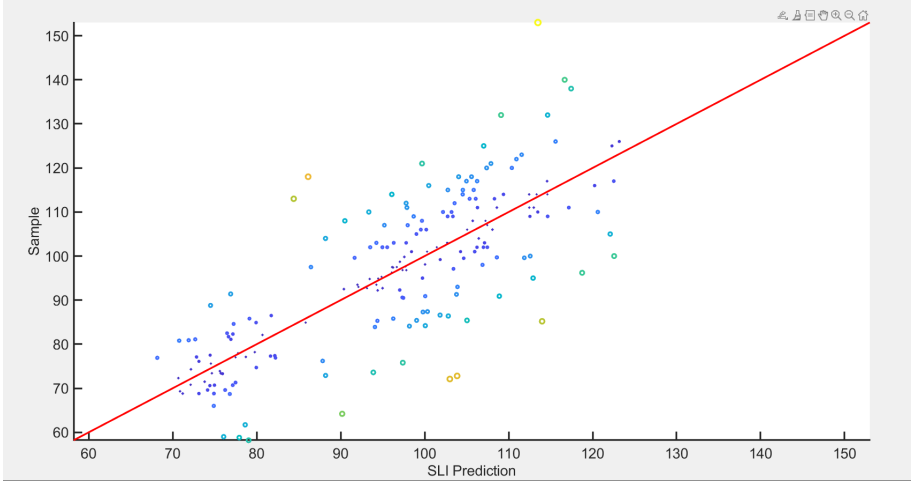

FIGURE 5.6: Original SLI predictions vs Real values using sample to sample precision matrix

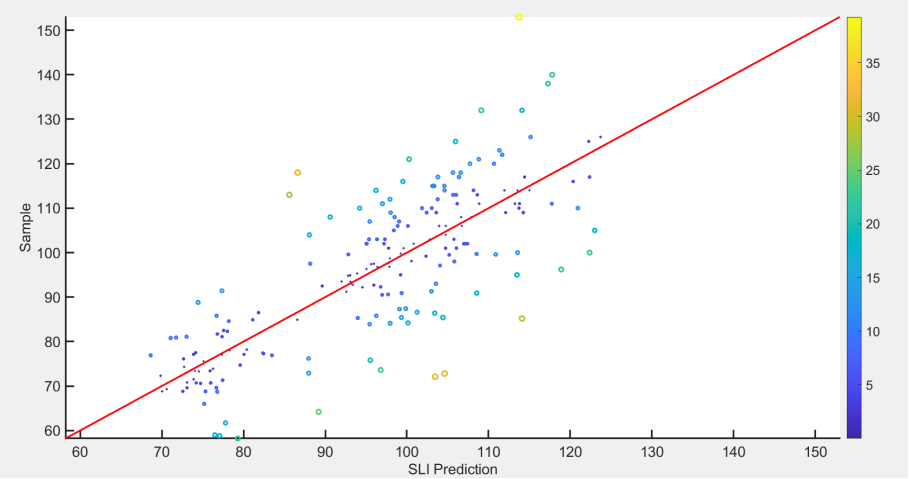

FIGURE 5.7: SLI GL predictions vs Real values using sample to sample precision matrix

In both cases there are some prediction errors but in most of the cases the error values are not greater than 1.0. However, the important thing is that the new model performs, if not better, the same with the original. The prediction errors are equal (in fact are a bit less), while the differences among the predicted and the sample points are not greater than the original. It is safe to say that from this first experiment for sample to sample, the new model seems to perform as well as the original one.

The bandwidths for gradient (h_1) and curvature (h_2), calculated in both cases are identical. This can be seen in the next visualizations:

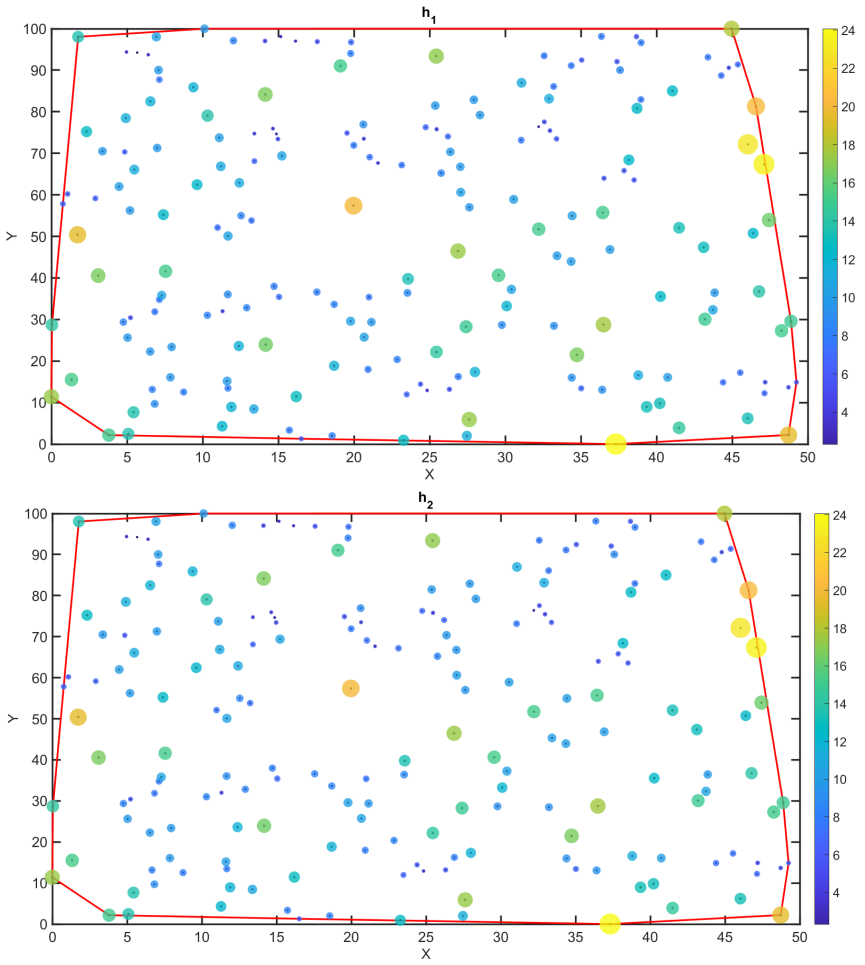

FIGURE 5.8: Bandwidths for gradient h1 (left) and curvature h2 (right) of the original SLI model

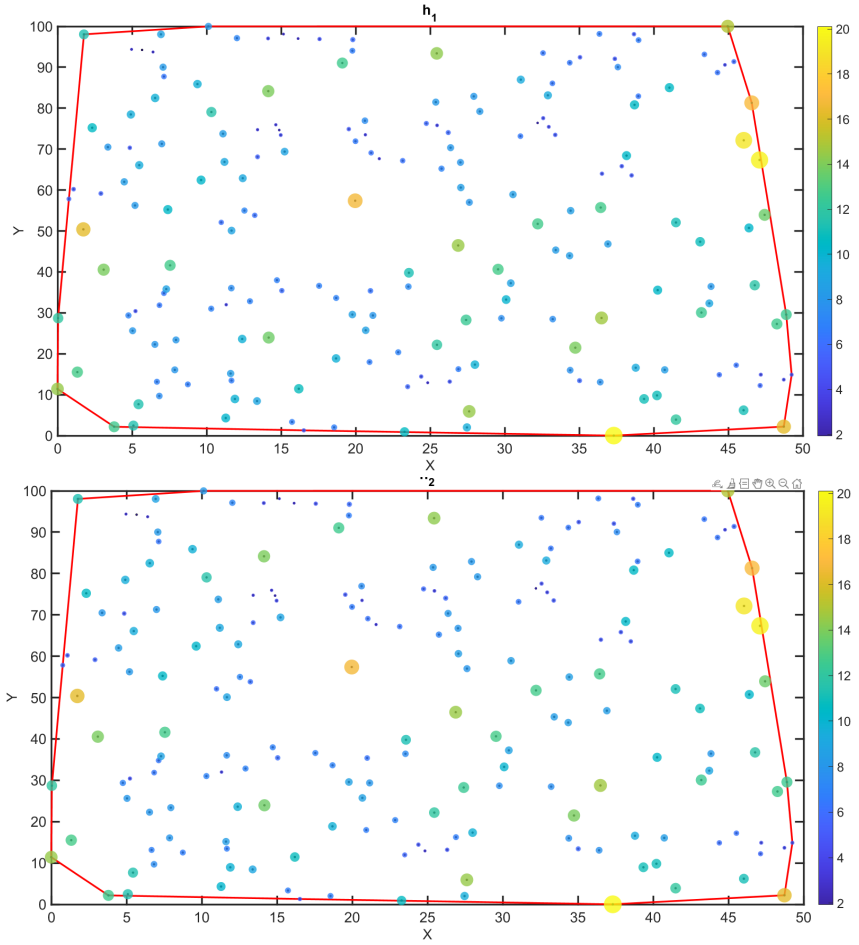

FIGURE 5.9: Bandwidths gradient h1 (left) and curvature h2 (right) of the original SLI model

The fact that in both models the bandwidths are identical, is the expected, because their calculation has nothing to do with the way the precision matrix is calculated, but with the distance matrix.

Predictions - Accuracy Validation

As mentioned above, the previous visualizations were for the better understanding of the experiment, and the validation of the model for the sample to sample precision matrix using leave one out cross validation. Now it is time to test the GL SLI to the real nature of the experiment, where using the sample points (in this experiment 200 sample points), the interpolation of the predictand points (in this experiment 808) will be made. The comparison of the 2 models will be through histograms, visualizations like the one of

the figure 5.5 and 5.6 and also a mapping using those interpolated values to test the smoothness.

Running both models the interpolated values were gained. It is important to note that the aim was to not reduce the accuracy of the SLI. In order, to see that this is achieved a histogram for both models was plotted, in which the deviation of the predicted to the real value is depicted with the frequency of the error. Both of them are provided next:

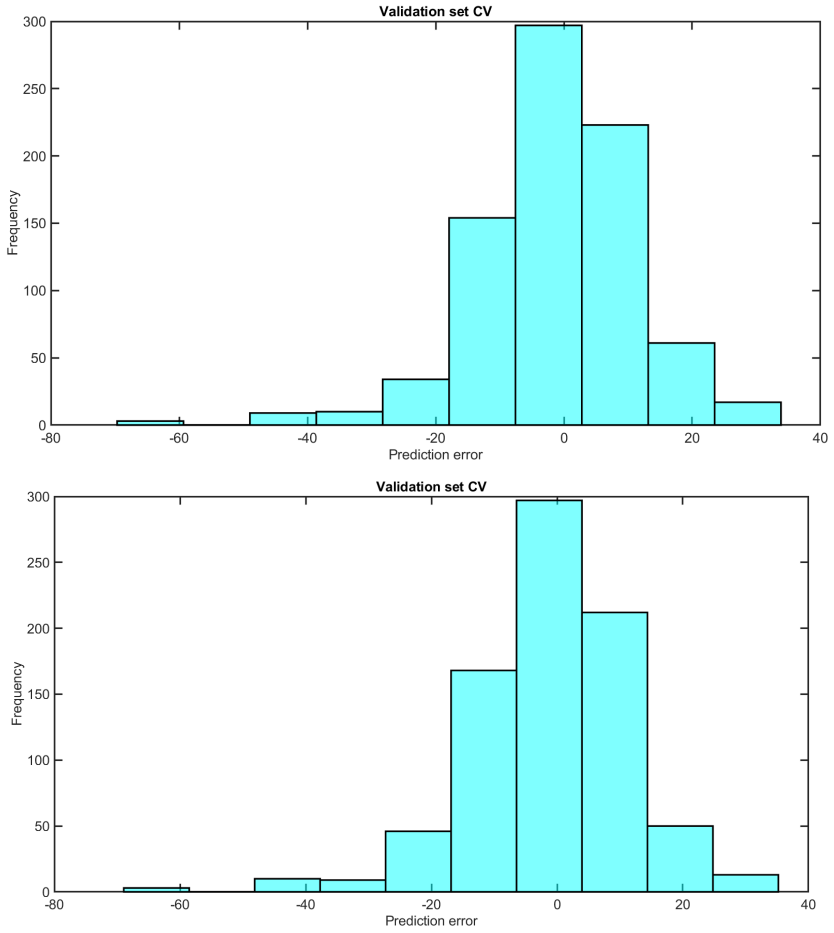

FIGURE 5.10: Histograms represent the Frequency-Error deviation of the SLI model original (left) and its GL version (right)

Considering the above, it can be observed that in both models the deviation near zero has the highest frequency while for prediction error greater than 20 or smaller than -20, the frequency is very low.

The efficiency of both models, the predicted accuracy and the errors are more understandable when depicted in a scatter plot. The scatter plots are given below:

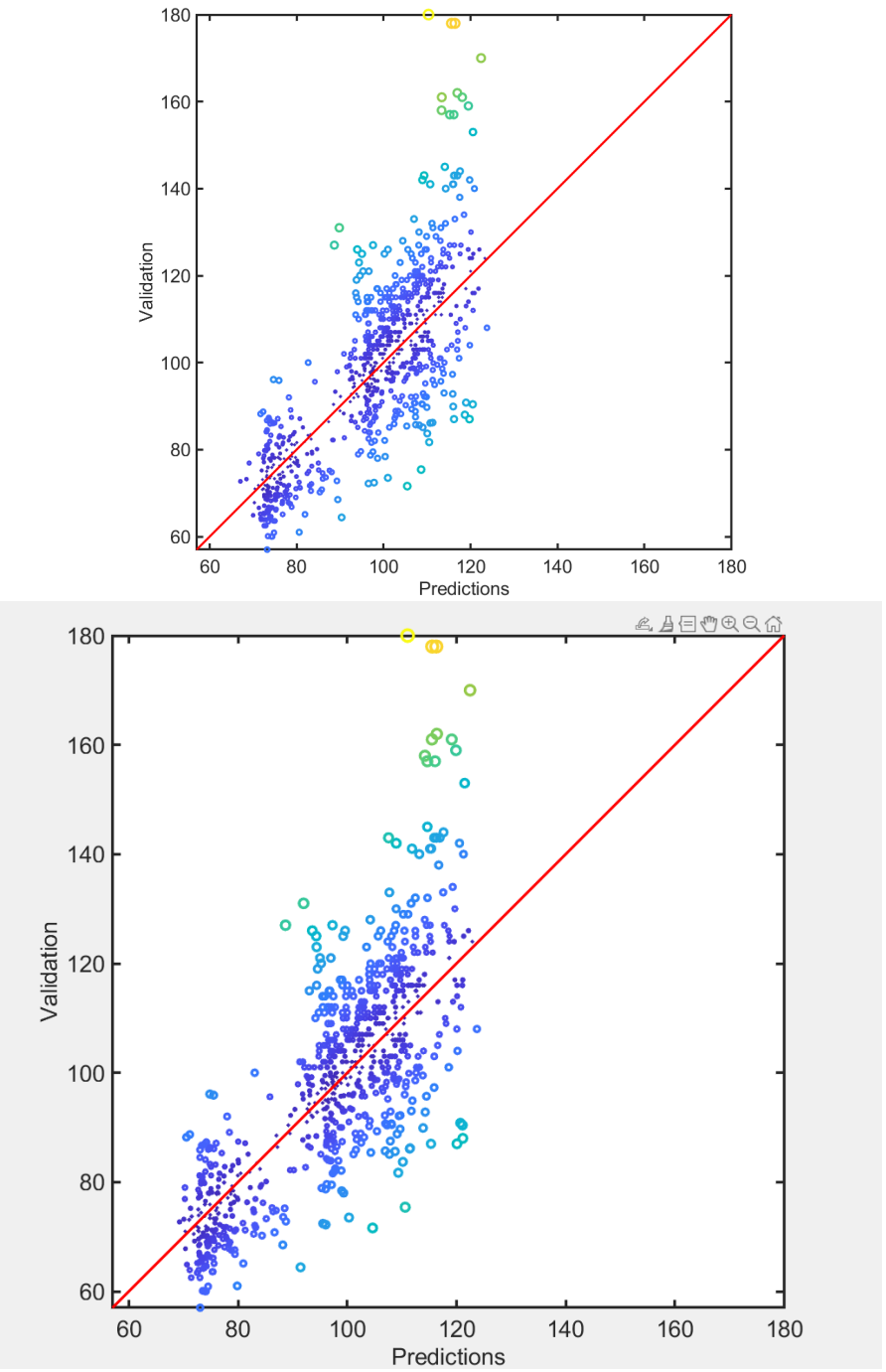

FIGURE 5.11: Scatter plots of the Predicted vs Real values of the SLI original (left) and SLI GL (right)

Most of the points are in general near the red line, while the deviation is not great in most of the cases. However as mentioned many times, the important thing is that the new model behaves the same (if not better) with the original one. Here the points have almost the same deviations in the same coordinates.

Lastly, it is useful to show in a histogram, where the predictions fall in comparison with the real values for the 2 models. The 2 plots are given in the following diagrams:

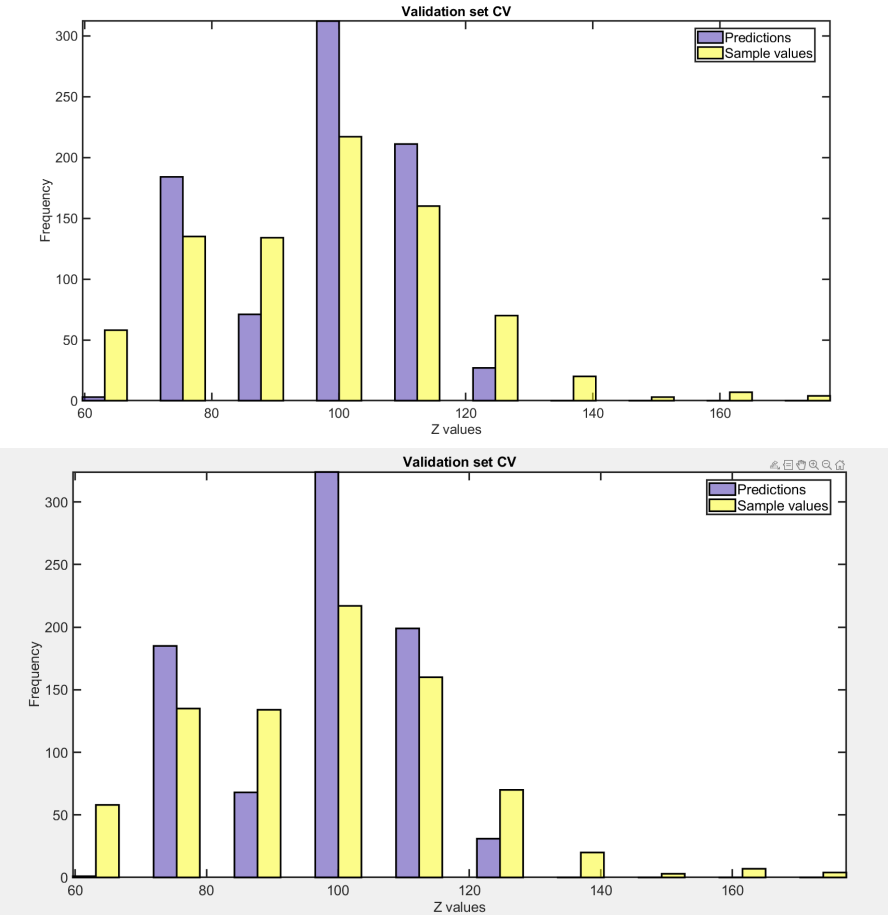

FIGURE 5.12: Histograms of the real vs predicted values and the frequency from the original model (left) and the GL version (right)

In both cases the models seems to have a weakness when it comes to predict really high or low values. Their efficiency however is almost identical, the errors are in the same points with the same deviations. So, it is easy to say that the new model not only doesn't have worse performance than the original one, but in many cases it behaves even better.

Generation of Interpolation Maps

Having the sample and the interpolated values occur from both the original and GL SLI models, the mapping of them can easily happen. The map is provided next:

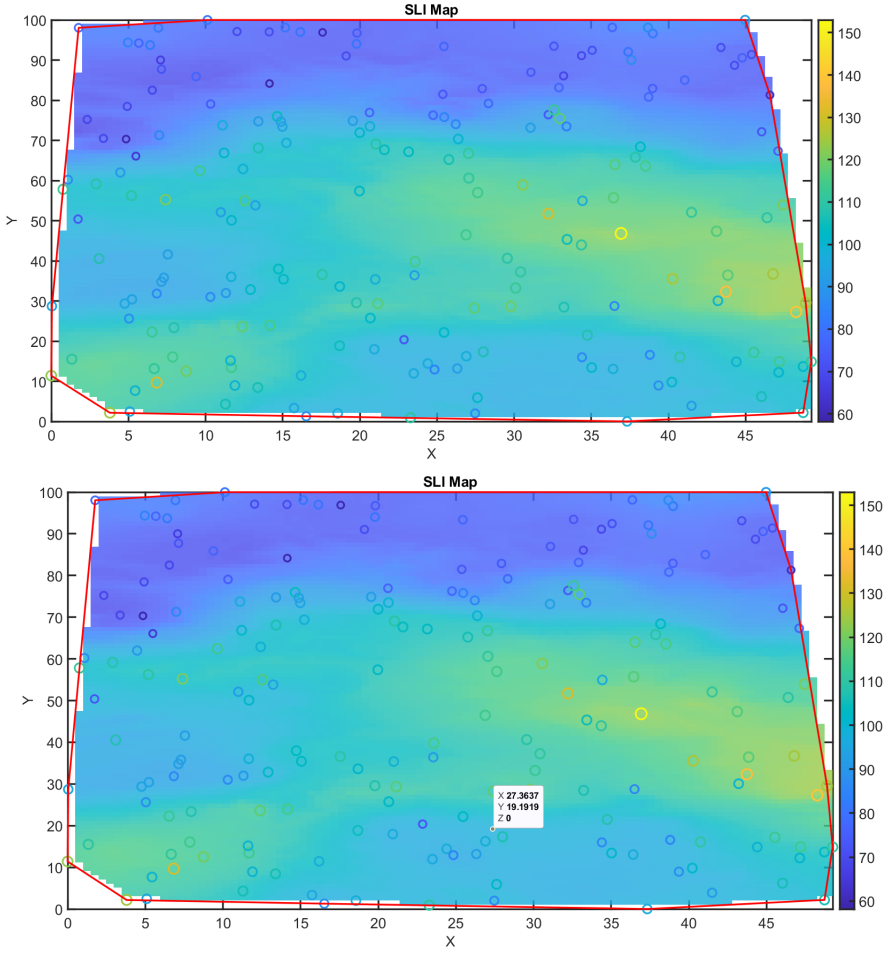

FIGURE 5.13: Map of the sample and interpolated values from original SLI (left) and the GL SLI (right)

The smoothing effect near the sample values, is more pronounced than that caused in the map of the figure 5.2. Of course, this has to do with the way of interpolation, r in the one of the figure 5.2 is generated by bilinear interpolation, while in the case of the maps depicted in the 5.12 the interpolation happens with the use of the SLI original and the GL SLI. It is worth to note also, that the GL SLI performs as well as the original one, a fact that can be observed from the smoothness.

Leave One Out (LVO) Cross Validation

A stability analysis by removing one sampling point at a time and determining the optimal SLI model using leave-one-out cross validation with that point removed, is conducted. The variation of the 2 models parameters are depicted in the next figures (fig. 5.13 for the original version and 5.14 for the GL version). Parameters α_1, α_2, and μ

are quite stable, whereas λ shows more variability. The spikes in the plots of the figure are exaggerated by using a narrow vertical range to better illustrate the parameter variability. For α_1, α_2, the maximum relative variation (with respect to the mean) ranges from a fraction of a thousandth (for α_1) to few thousandths (for α_2); μ shows stronger variations, whereas the strongest variation is exhibited by λ, since the latter is a scaling factor that determines the overall energy of the ensemble of points and compensates for variations in the other parameters. It is possible that the parameter variations exhibited in the plots are, at least partially due to extremely slow variation of the cost function over a region of the parameter space, a condition also observed in maximum likelihood estimation of spatial models with Matérn covariance. This slow variation implies quasi-degeneracy of the parameter vector; the quasi-degeneracy implies that vectors which are very far in parameter space may lead to very similar cost function values. More recently, the difficulties involved in nonlinear fits of multi-parametric models to data have been investigated.

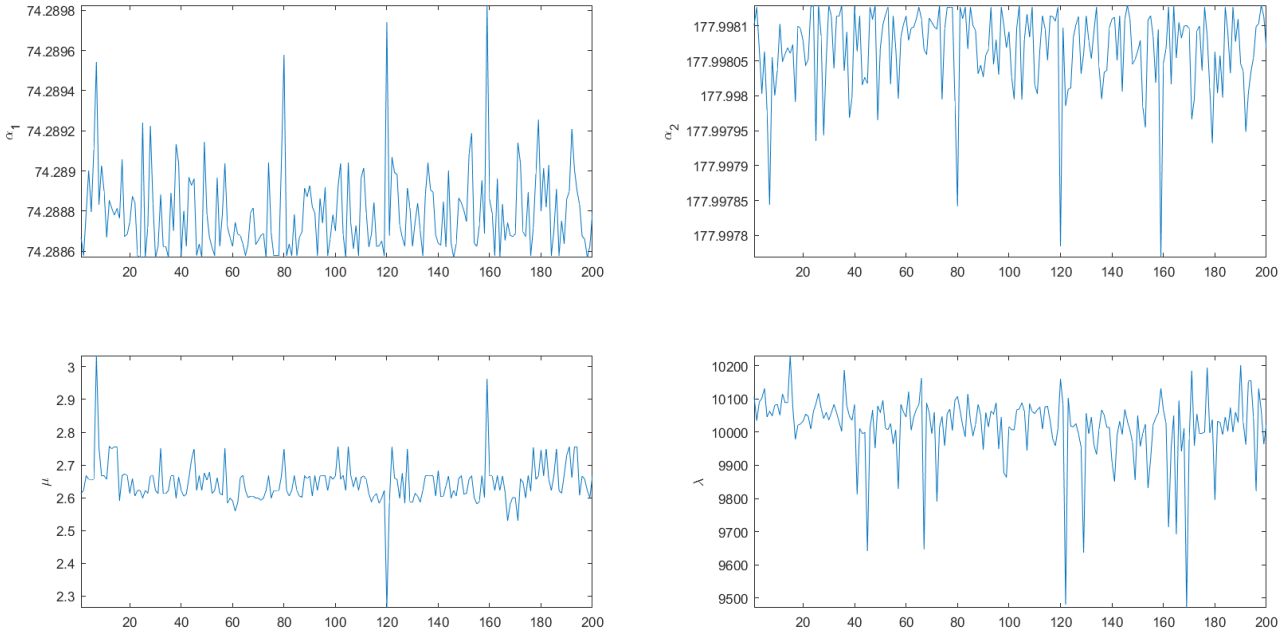

FIGURE 5.14: Parameter Variations of original SLI model

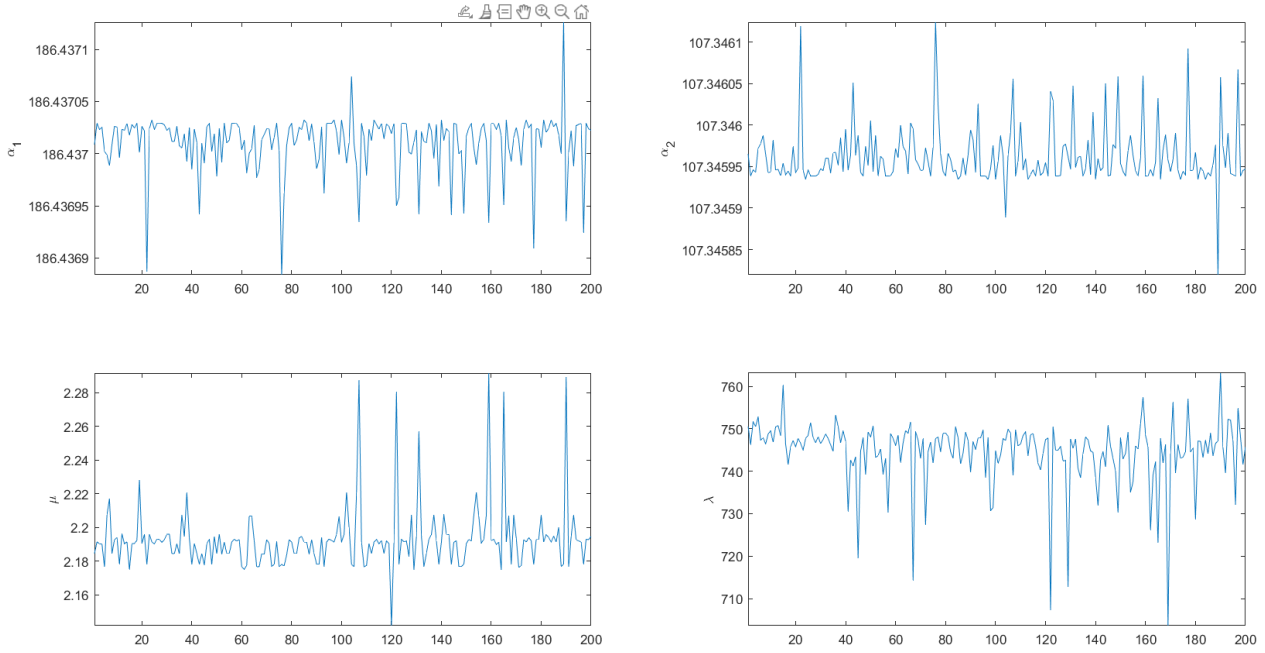

FIGURE 5.15: Parameter Variation of SLI GL

By focusing to the figures 5.13 and 5.14 some observations can be made. In both models the parameters α_1, α_2, and μ are quite stable (the spikes show differences in the thousandth), while λ has a more noticeable difference per iteration. The values from model to model may differ a bit, but it is something perfectly normal cause of the different precision matrix. In addition, Plots of LVO cross validation SLI parameter estimates and of SLI energy versus the value of the sampling point removed is presented for both models in figures 5.15 and 5.16.

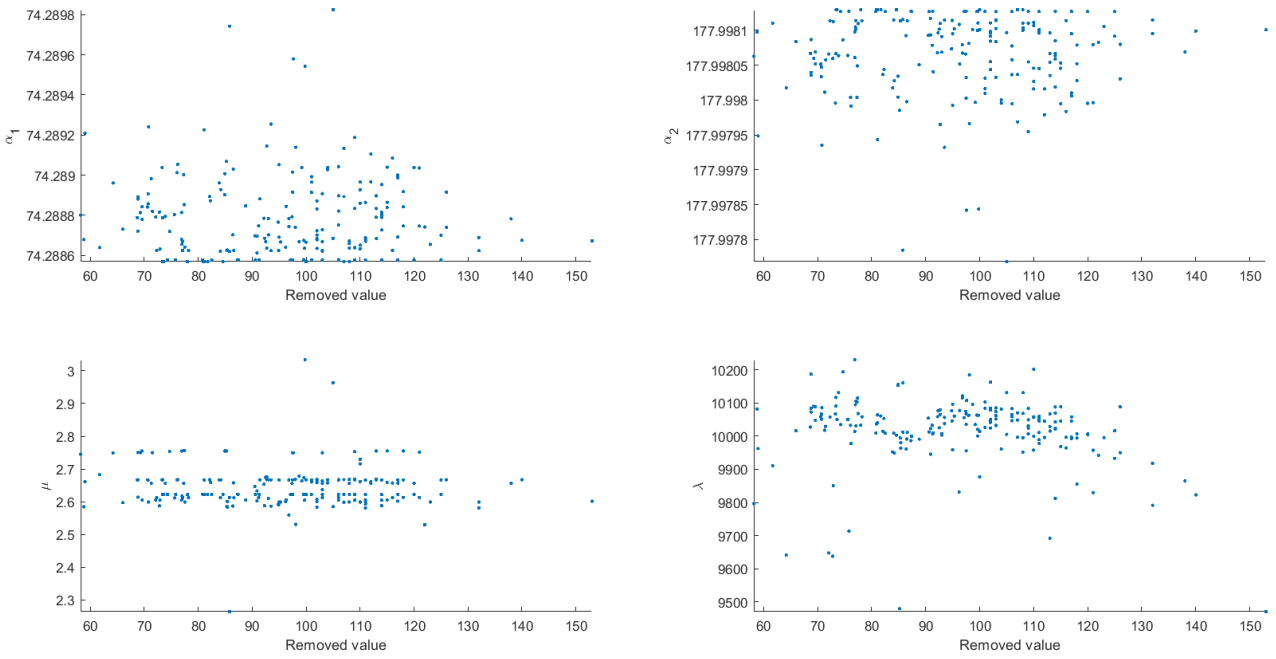

FIGURE 5.16: Parameter Variations - Removed values of original SLI model

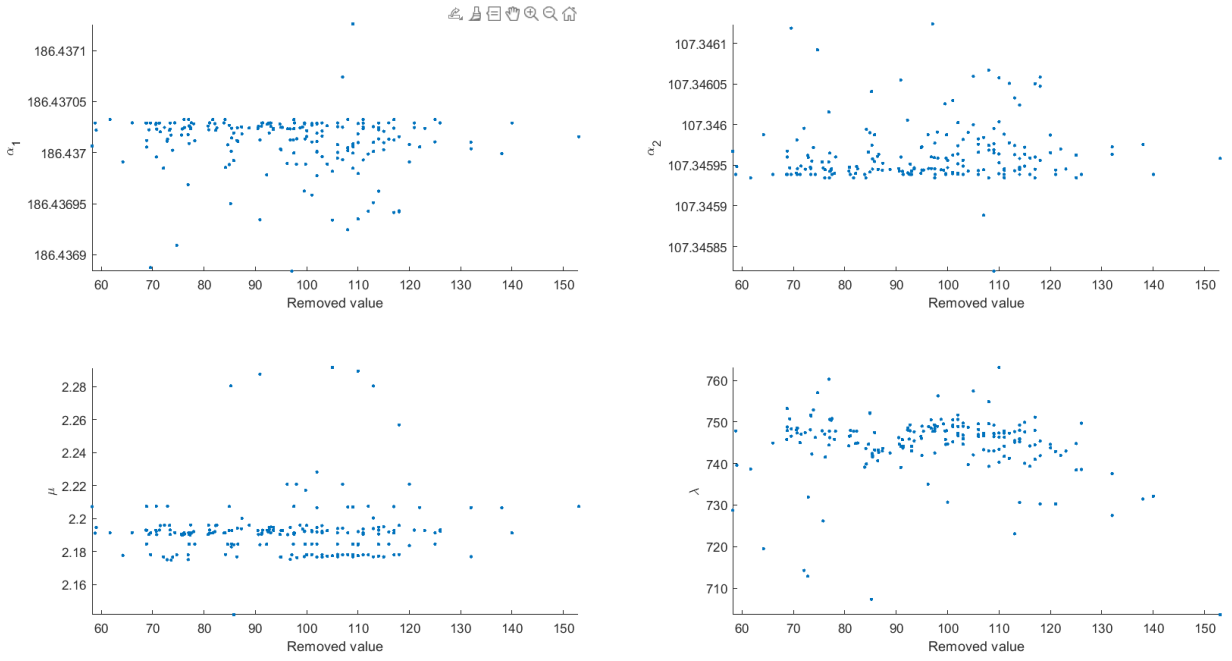

FIGURE 5.17: Parameter Variations - Removed values of SLI GL

It seems that in the case of the GL version the parameters seem a lot more stable for every value removed. In the case of the original model it can be seen that there are very scattered, while in the new version are a lot more stable.

The validation of the GL SLI model was initially conducted using cross-validation. To enhance the robustness of the validation process, Leave-One-Out (LVO) cross-validation is now being employed. This method involves iterative running the model N times, where N represents the number of sample points in the dataset. During each iteration, one sample point is sequentially withheld from the training set (locs) and designated as the point to be predicted (plocs). Consequently, each iteration focuses on predicting a single point, resulting in a sample-to-predict precision matrix of dimensions 1×199, where each time a different point is removed from the set. Following this iterative process, SLI parameter estimates are computed for each iteration, and their visualizations illustrating their variation are provided above. Furthermore, to assess the model's performance post LVO cross-validation, several diagnostic plots are presented. These include a histogram depicting the distribution of prediction errors and their corresponding frequencies, a scatter plot illustrating predicted values versus actual values, and histograms displaying the distribution of z-values for both real and predicted points along with their frequencies. To facilitate comparative analysis, visualizations of the original model are also included, enabling a comprehensive visual assessment of model performance.

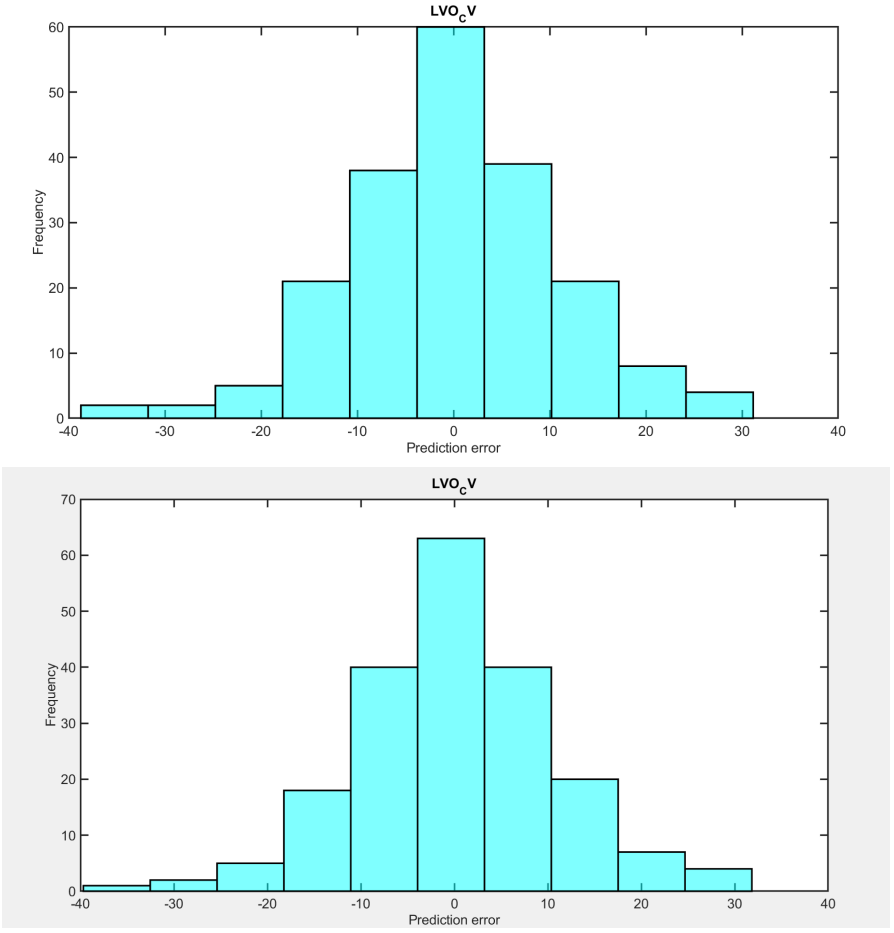

FIGURE 5.18: Histogram of the Prediction errors and their frequencies after the LVO cross validation for the original model (left) and for the GL version (right)

Observing figure 5.18 it can be noted that the original model's error distribution is centered around zero, indicating a reasonable fit; however, it shows a broader spread, with more instances of higher errors. In contrast, the GL version of the model exhibits a tighter error distribution around zero, suggesting enhanced accuracy and fewer high-magnitude errors.

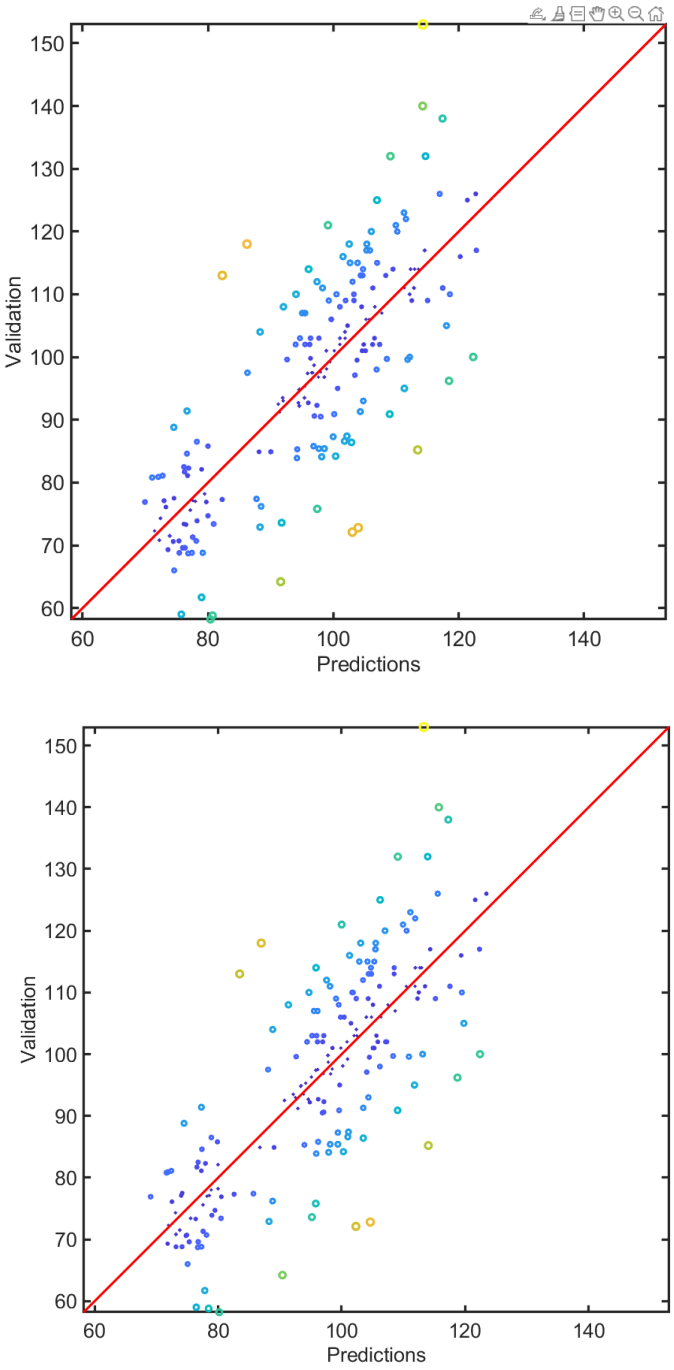

FIGURE 5.19: Scatter Plot: Predicted Vs Actual Values with LVO Cross Validation for the original model (left) and for the GL version (right)

The scatter plots comparing predicted and actual values (figure 5.19) indicate that both models perform (almost) similarly. The points are closely aligned along the line of equality in both plots, demonstrating that the GL model's predictions are comparable to those of the original model. This similarity suggests that while other metrics show improvements, the overall prediction accuracy remains consistent between the two models.

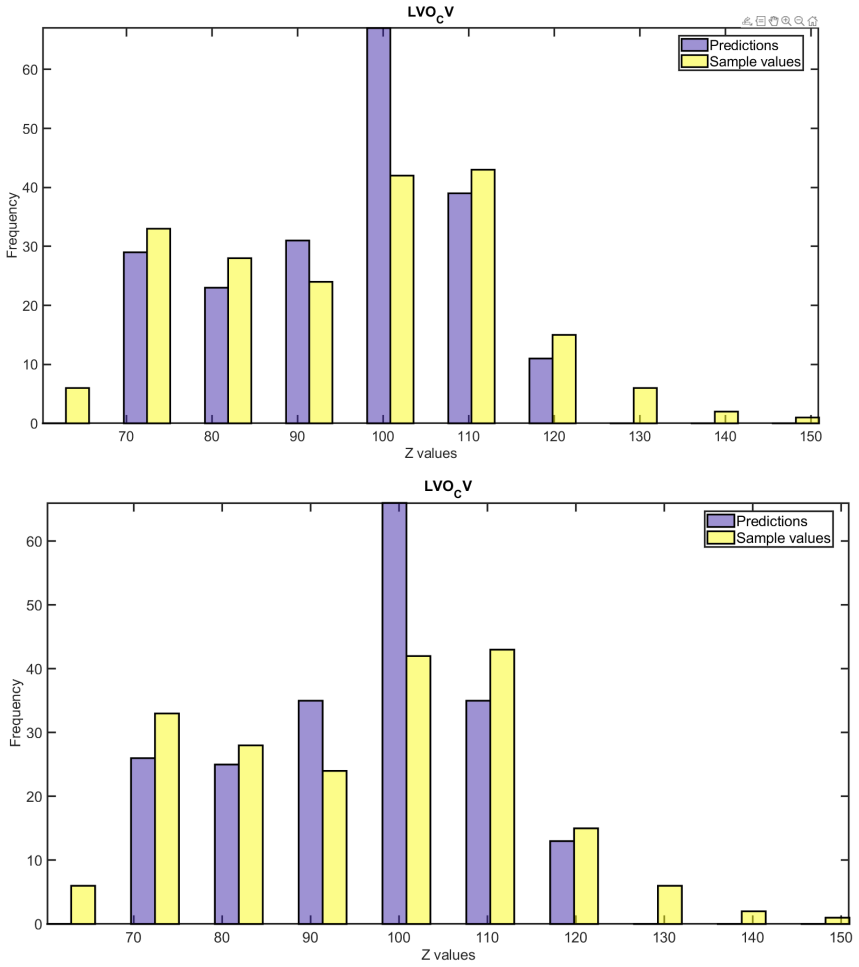

FIGURE 5.20: Histogram of the Real and Predicted values and the frequencies after the LVO cross validation for the original model (left) and for the GL version (right)

The histograms of real versus predicted values (figure 5.20) reveal that both models exhibit some weaknesses in predicting higher z-values. In certain instances, the predicted values are closer to the real ones in the original model, while in other cases, the GL model shows improved predictions. This variability suggests that each model has strengths in different areas, and neither model consistently outperforms the other across all z-value ranges. Of course, the observed differences do not conclusively indicate that one model outperforms the other.

5.2.2 4D - Synthetic data from deformed Gaussian

While the new GL SLI model demonstrates promising performance in the 2D experiment, the original SLI model was designed to operate effectively across any number of dimensions. Therefore, it is imperative to evaluate the GL version in a higher-dimensional

context to verify that it performs comparably to the original model. To this end, a four-dimensional synthetic dataset generated from a deformed Gaussian distribution is utilized for this experiment. To better understand the experiment, two different visualizations are presented: 2D projections of the samples and the distributions of the sample and prediction point values. The experiment is repeated by adding Gaussian noise to the dataset and the range of noise-free z values is 1.0137.

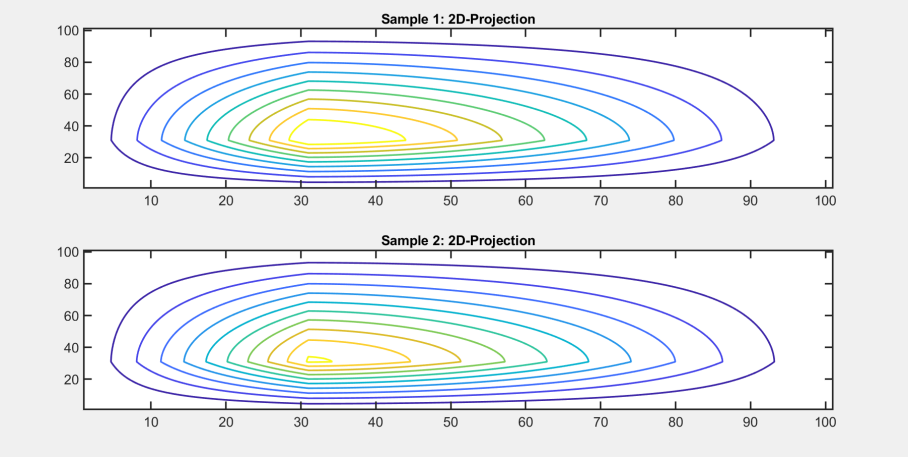

FIGURE 5.21: 2D Projections of the 4D experiment

These projections are obtained by selecting pairs of dimensions and visualizing their relationships in two-dimensional plots. Each contour plot represents the projection of the four-dimensional synthetic data into a two-dimensional space, showing how the data points are distributed across selected pairs of dimensions. The contour lines indicate regions of similar values, providing insights into the data's underlying patterns and distributions.

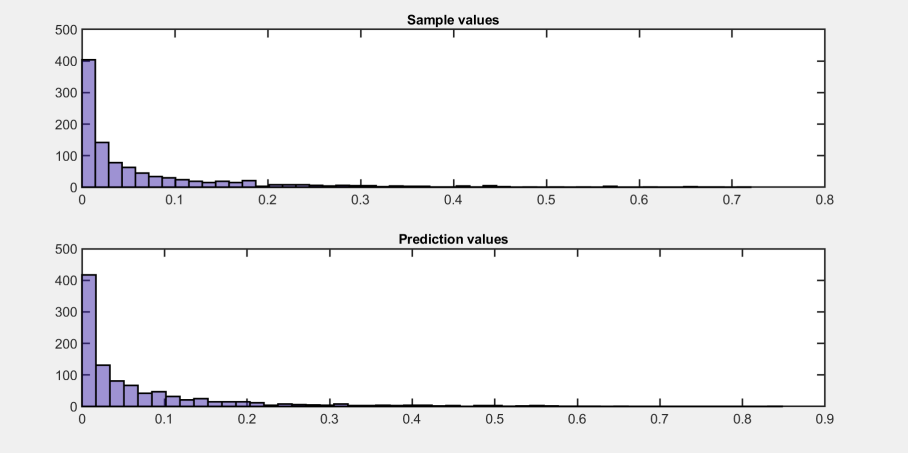

FIGURE 5.22: Distributions of sample-predicted point values of 4D Experiment

In addition, the following function is considered:

$$\chi(\mathbf{s}) = Ae^{-2\|\mathbf{s}-\mathbf{a}\|} \prod_{i=1}^{4} s_i(1-s_i), \tag{5.2}$$

where $A = 500$ and $\mathbf{a} = (0.3, 0.3, 0.3, 0.3)$ are defined over the four-dimensional cube with unit length edges, i.e., for $\mathbf{s} \in [0,1]^4$. The function is sampled at $N = 1000$ randomly selected points over the unit cube, and a validation set of $N = 1000$ points is generated through random selection.

The SLI model's optimal parameters for the quadratic kernel with $k = 2$ are determined as $\alpha_1 \approx 10.12$, $\alpha_2 \approx 25.04$, $\mu \approx 1.64$, and $\lambda \approx 0.0193$, starting with initial values $\alpha_1 = 10$, $\alpha_2 = 25$, $\mu = 3$. Similar cross-validation performance results are achieved with different initial conditions leading to various local optima.

The sparse structure of the precision matrices, are depicted in Figure 5.23 for both the original SLI and the GL version, and display the logarithm of the absolute value.

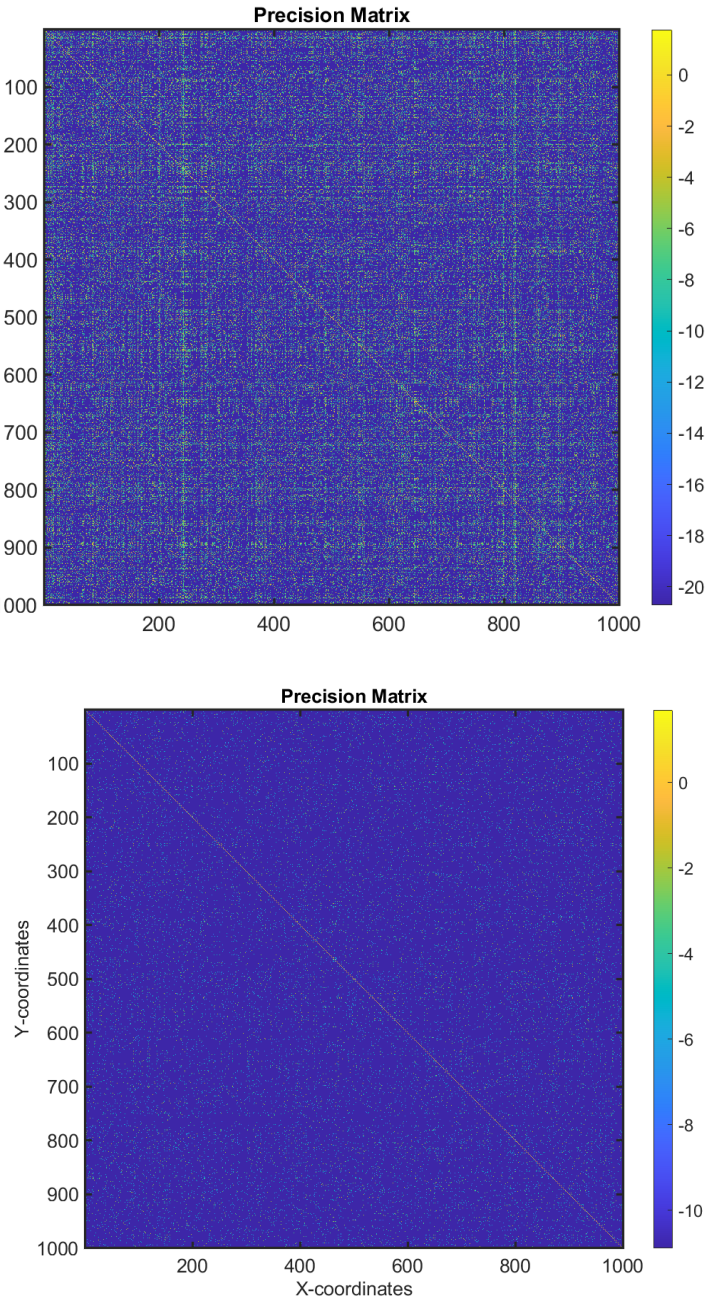

FIGURE 5.23: Precision Matrix (sample to sample) of the 4D experiment using the original SLI (left) and the GL Version (right)

The main differences between the two models are the range and distribution of precision values. The GL model has a narrower range centered around 10, with more uniform values and rare higher values. In contrast, the original SLI model exhibits a broader range, from 0 to 20, with most values below 15 and rare instances of higher values. The GL model maintains a relatively uniform precision value across most samples, with occasional higher values, while the original SLI model captures a broader spectrum of

relationships between samples, from very weak to moderately strong. The values in both cases in the diagonal are near 1 or less and as for the sparsity, the matrices are more sparse than those of the previous experiment. In the case of the SLI original the non zero elements are $183,378$ out of $1,000,000$ with sparsity score of 81.7% and in the GL-SLI are $44,038$ out of $1,000,000$ with sparsity score of 95.6%. In conclusion, the GL model maintains more uniform precision values with occasional higher values, suggesting a consistent approach to relationship modeling. From the other hand, the original SLI model, with its broader range and occasional higher values, indicates a more varied approach to capturing relationships.

The scatter plots of the validation values versus the respective SLI predictions are shown in Figure 5.24. Although, it may seem that the GL-version demonstrates poorer agreement of points, it is not significantly poorer than the one of the original version (performs almost the same).

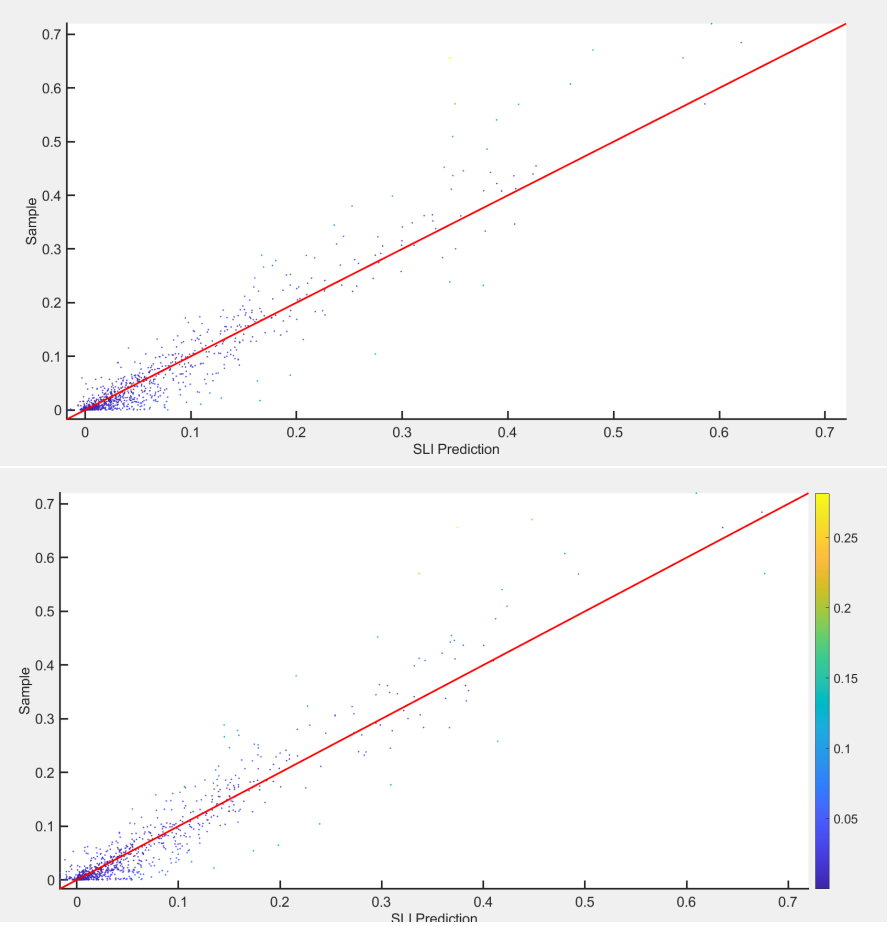

FIGURE 5.24: Scatter plots of SLI original (left) and GL version (right) predictions versus the respective values of the validation set (Precision matrix used: Sample to Sample)

Observing the scatter plots it can be verified that both models perform better than the case of the 2D. However, the important thing is that they perform almost the same and that the GL version doesn't reduce significantly the performance. Additionally, those scatter plots are a result of the model which is using the sample to sample precision matrix. Thus, it is wise to validate their performances by using the model as it should be: Predict the set of the values using the sample set by using the precision matrix sample to predict. The scatter plots of the latter case are given below:

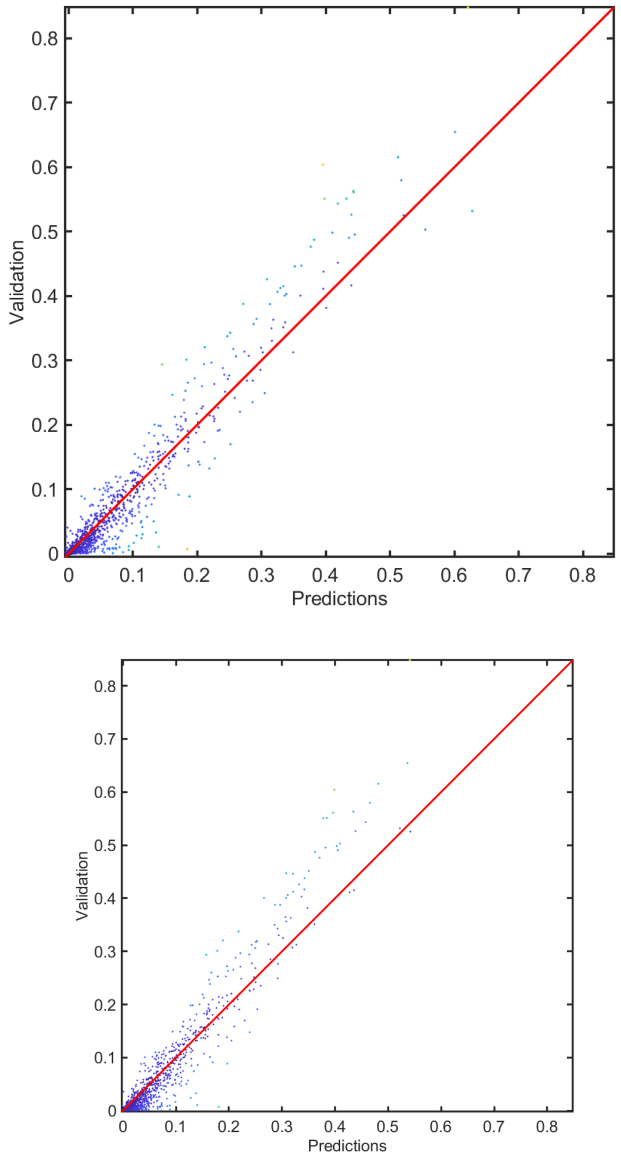

FIGURE 5.25: Scatter plots of SLI original (left) and GL version (right) predictions versus the respective values of the validation set (Precision matrix used: Sample to Predict)

In both cases, there is a small deviation from the line of equality (red line). It can be

observed that the allignment in the case of the original model is slightly better than the one of the GL-version but, this deviation is very small with not significant preformance reduction. This can be seen better in the following histograms.

As mentioned previously, it is important to visualize the cross validation results from both models to have a better comparison of their performance. More specifically, 4 histograms are presented (2 for each model), in order to see the validation errors and their frequencies (figure 5.26) and the predicted values versus the actual values with the frequencies (figure 5.27). The visualizations are presented next:

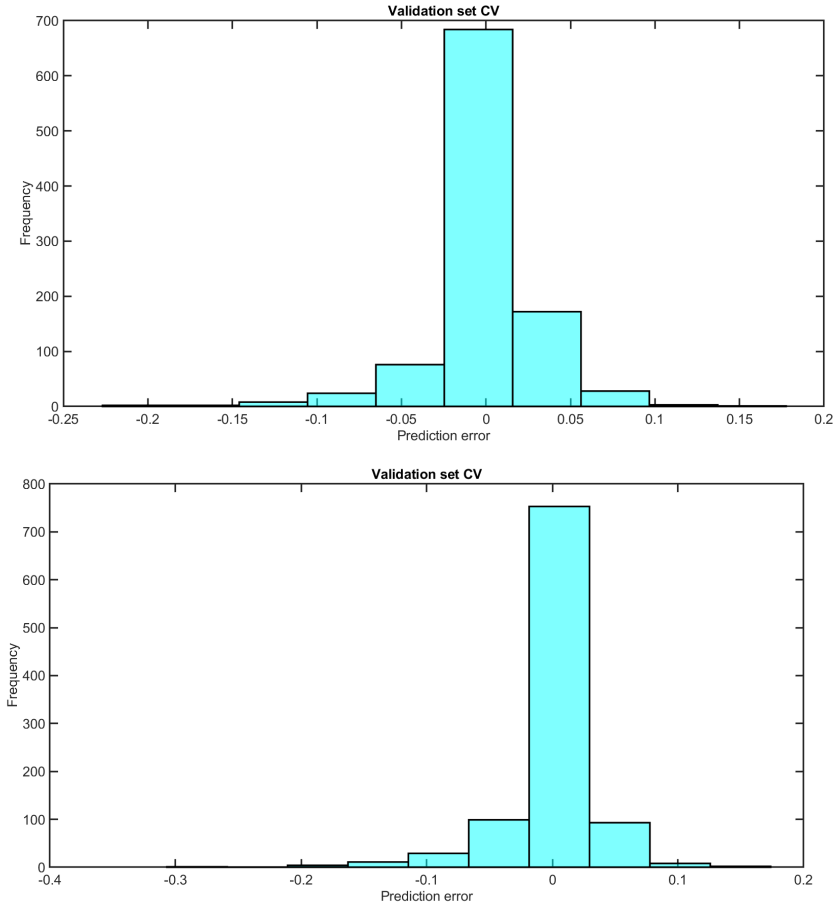

FIGURE 5.26: Histogram of the Prediction Errors and their frequency of SLI original (left) and GL version (right)

In both cases the distribution of the prediction errors is centered around zero. By comparing the 2 histograms, it can be noted that the 0 error in the case of the original model is a bit less frequent. In the case of the GL-version the errors are equally split between -0.1 and the 0.05 while it has a bit more frequent errors in 0.1. As for the rest of the errors, in both models the range of the error values relies from -0.2 to 0.15.

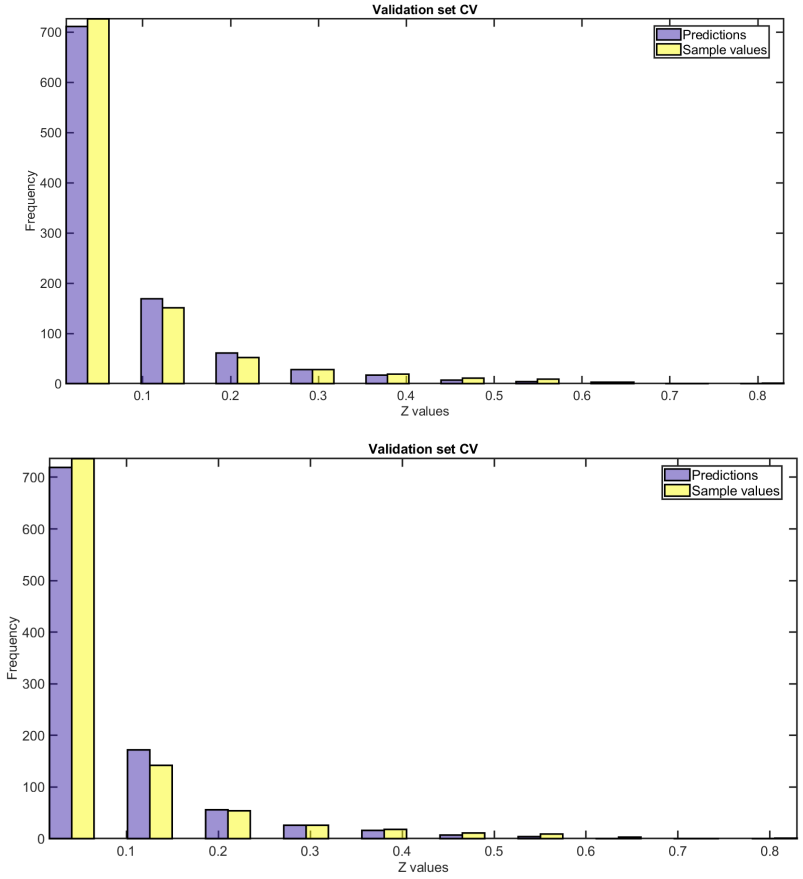

FIGURE 5.27: Histogram of the real vs Predicted values after cross validation of SLI original (left) and GL version (right)

It can be seen that in contrast with the experiment of the 2 Dimensions the predicted values (a small range of them) in both cases are (almost) perfectly aligned with the sample values (a small range of them). In the case of the GL SLI model there is an undeprediction in the values that are near zero which result an overprediction in the values near 0.1, but the same pattern is observed in the original model too. This indicates that the performance has not been reduced, and the value differences has to do with the core of the models.

5.3 Comparative Analysis - Error Metrics

While the above visualizations can indeed show the wanted results, and through comparison the validation of the new model, it is important to show the error metrics of the new model and also compare them with the original one. More specifically, the error metrics that are going to be provided are for the above 2 experiments. The metrics that

are used are the **Mean Error (ME)**, **Mean Absolute Error (MAE), Mean Absolute Relative Error (MARE), Root Mean Square Error (RMSE), Pearson Correlation Coefficient (RP)** to measure linear correlation between two sets of data and **Spearman's Rank Correlation Coefficient (RS)** to measure the strength and direction of association between two ranked variables . The formulas of each metric are given in the following table:

Metric	**Formula**
Mean Error (ME)	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$
Mean Absolute Error (MAE)	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
Mean Absolute Relative Error (MARE)	$\frac{1}{n}\sum_{i=1}^{n}\left\lvert\frac{y_i-\hat{y}_i}{y_i}\right\rvert$
Root Mean Square Error (RMSE)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
Pearson Correlation Coefficient (RP)	$\frac{\sum_{i=1}^{n}(y_i-\bar{y})(\hat{y}_i-\bar{\hat{y}})}{\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2\sum_{i=1}^{n}(\hat{y}_i-\bar{\hat{y}})^2}}$
Spearman's Rank Correlation Coefficient (RS)	$1 - \frac{6\sum_{i=1}^{n}d_i^2}{n(n^2-1)}$

TABLE 5.1: Metrics and their formulas

Starting from the 2D SIC 2004 Radioactivity data the error metrics from the original model in comparison with the one with the GL implementation can be seen in the next board:

	ME	**MAE**	**MARE**	**RMSE**	**RP**	**RS**
SLI Original	−1.29	9.28	0.09	12.60	0.78	0.76
SLI GL	−1.28	9.25	0.09	12.54	0.78	0.76

TABLE 5.2: Metrics Comparison between SLI Original and SLI GL for the 2D SIC 2004 Radioactivity data

It is important to test those metrics in higher dimensions too. In order to do that the experiment from above in 4 dimensions about the Synthetic data from deformed Gaussian is used. The table containing the metrics from both models is provided next:

	ME	**MAE**	**MARE**	**RMSE**	**RP**	**RS**
SLI Original	0.001	0.020	5.445	0.031	0.962	0.901
SLI GL	0.001	0.020	9.2839	0.034	0.961	0.904

TABLE 5.3: Metrics Comparison between SLI Original and SLI GL for the 4D Synthetic data from deformed Gaussian

In both experiments the error metrics between the original SLI model and the one with the Graph Laplacian implementation are almost identical. Also it is worth to note that there is notable linear correlation between the 2 sets in both cases. The strength of those set measured with Spearman's rank correlation Coefficient, also seems to not change using the GL version of the SLI.

In order to have a global view a table with the above metrics from different models using Machine Learning or Geostatistics for the first experiment, is provided below (Dubois (2005)):

	ME	**MAE**	**RMSE**	**RP**
Neural Network	−1.25	9.40	12.59	0.78
SVM	−0.04	9.22	12.47	0.79
GeoStats	−1.32	9.06	12.43	0.79
Splines	1.60	9.30	12.60	0.78

TABLE 5.4: Summary table with the results of the SIC2004 experiment using Neural Networks, Support Vector Machines, Geostatistics techniques and more

Chapter 6

Conclusion & Future Work

6.1 Conclusion

The SLI model presented above provides a bridge between geostatistics and machine learning. It is based on an exponential joint density which involves an energy functional with an explicit precision (inverse covariance) matrix. The latter is constructed by superimposing network sub-matrices that implement local interactions between neighboring field values in terms of kernel functions. The algorithmic complexity of SLI missing value estimation scales linearly with the sample size except for a global $O(N^2)$ term which is, however, computed once for all the prediction points. Hence, the leave-one-out cross-validation approach can be used to efficiently infer the SLI model parameters.

Though the SLI model seems to be very stable with good performance, there are cases in which it can be very poor. More specifically, assumptions for the positive (semi) definiteness of the precision matrix have been made. There are datasets that may not follow these assumptions and that's why an updated version of the SLI was presented in this work. The precision matrix of the model is now built using the Graph Laplacian (specifically the second order of the graph Laplacian for the curvature), to ensure the postive (semi) definiteness of the precision matrix, and also by optimizing the complexity of the model. In future it is possible to eliminate the network matrices for the calculation of the precision matrices and use the graph Laplacian for the computation of the gradient too. In conclusion, the new version, performs as well as the original model, but by ensuring the assumptions made in the first case are now holding for every kind of data.

6.2 Future Work

The next phase of research will explore modifications to the energy functional H of the Stochastic Local Interaction (SLI) model by also replacing the term S_1 , which relates to the gradient, with an adaptation involving the Laplacian. This adjustment aims to refine the model's responsiveness to spatial variations by integrating a more comprehensive representation of spatial derivatives, and eliminate the calculation of the network matrices. The anticipated modification to H would not only involve redefining the gradient aspects but also reassessing the balance of local versus global interactions within the data. The prospective modification is expected to enhance the model's predictive accuracy and adaptability, particularly in complex spatial environments where traditional gradient measures might fall short. This exploration will likely open new pathways for integrating more sophisticated mathematical tools into spatial analysis, pushing the boundaries of how we understand and interact with spatial data.

Gradient through Laplacian

Gradients can indeed be calculated using the Laplace operator, especially in the context of partial differential equations and vector calculus. The case that is about to be examined, is to use the incidence matrix. First of all, let's define the incidence matrix:

An incidence matrix is a logical matrix that shows the relationship between two classes of objects, usually called an incidence relation. If the first class is X and the second is Y, the matrix has one row for each element of X and one column for each element of Y. The entry in row x and column y is 1 if x and y are related (called incident in this context) and 0 if they are not. In graph theory, it is different to an adjacency matrix, which encodes the relation of vertex-vertex pairs. In graph theory, an undirected graph has two kinds of incidence matrices: unoriented and oriented.

The *unoriented incidence matrix* (or simply *incidence matrix*) of an undirected graph is an $n \times m$ matrix B, where n and m are the numbers of vertices and edges respectively, such that

$$B_{ij} = \begin{cases} 1 & \text{if vertex } v_i \text{ is incident with edge } e_j, \\ 0 & \text{otherwise.} \end{cases}.$$

Having the incidence matrix of a graph, the Laplacian Matrix or Graph Laplacian can be calculated. Specifically, the latter is obtained from the oriented incidence matrix $\mathbf{B}(\mathbf{G})$ by the formula, $\mathbf{L}(\mathbf{G}) = \mathbf{B}(\mathbf{G})\mathbf{B}(\mathbf{G})^{\mathbf{T}}$, where $\mathbf{G}$ the graph of case.
While the traditional gradient is defined for continuous scalar fields, on graphs, it can be defined as the differences in values between connected nodes. This is often termed as the "graph gradient"or the "discrete gradient".
For a scalar function f defined on the nodes of the graph, the gradient can be represented on the edges of the graph. The gradient on the edge (i, j) can be thought of as the difference $f_j - f_i$.

In matrix terms, this difference can be encoded using the incidence matrix $\mathbf{B}$ of the graph, where $\mathbf{B}_{ei} = 1$ if edge e starts at node i and $\mathbf{B}_{ei} = -1$ if it ends at node i. For an undirected graph, you can define $\mathbf{B}$ as above.

In conclusion, the idea is, that having the graph Laplacian calculated, you can reach to the incidence matrix $\mathbf{B}$, using the formula presented above. Then, having the incidence matrix of the graph, the differences across edges for a scalar function defined on the nodes of the graph can be computed. This $\mathbf{B}f$ gives you a vector of differences corresponding to each edge in the graph, which are analogous to the gradient in continuous domains.

Bibliography

D. Bertsekas, J. T. (2008). *Introduction to probability*. Athena Scientific, USA.

Dionissios T. Hristopulos, V. D. A. (2020). Stochastic local interaction model with sparse precision matrix for space–time interpolation. *Spatial Statistics*, 73(4):1–15.

Dubois, G. (2005). *Automatic mapping algorithms for routine and emergency monitoring data*. Institute for Environment and Sustainability.

Eduardo Pavez, A. O. (2016). Generalized laplacian precision matrix estimation for graph signal processing. 25:1–5.

Horaud, R. (2015). A short tutorial on graph laplacians, laplacian embedding, and spectral clustering. *INRIA Grenoble Rhone-Alpes*, 25:1–43.

Hristopulos, D. T. (2015). Stochastic local interaction (sli) model: Bridging machine learning and geostatistics. *Computers and Geosciences*, 25:1–12.

Hristopulos, D. T. (2020). *Random Fields for Spatial Data Modeling: A Primer for Scientists and Engineers*. Springer, Dordrecht, the Netherlands.

Hristopulos, D. T. (2022). Boltzmann–gibbs random fields with mesh-free precision operators based on smoothed particle hydrodynamics. 25:1–24.

Lindgren, F. and Rue, H. (2011). An explicit link between Gaussian fields and Gaussian markov random fields: The SPDE approach. *Journal of the Royal Statistical Society, Series B*, 73(4):423–498.

Michael McCartney, M. H. . W. P. (2020). Comparison of machine learning algorithms in the interpolation and extrapolation of flame describing functions. *Engeeniring for Gas tourbines and Power*, 25:1–10.

Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields: Theory and Applications*. Chapman and Hall/CRC.

Sandra De Iaco, Dionissios T. Hristopulos, G. L. (2022). Special issue: Geostatistics and machine learning. 25:1–7.

Williams, C. E. R. . C. K. I. (2006). *Gaussian Processes for Machine Learning*. the MIT Press, Cambridge, Massachusetts London, England.