Bayesian inference for 4D spatio-temporal models using Integrated nested Laplacian approximations

Michal Sudwoj
msudwoj@student.ethz.ch

13th January 2023

Advisors:
Prof. Dr. Olaf Schenk
cand. PhD. Lisa Gaedke-Merzhäuser

Abstract

The discovery of Monte Carlo methods some 80 years ago spurred a renewed interest in Bayesian inference in the 1990s. Despite many advances in the field, large models are hampered by the *curse of dimensionality*, and cannot be efficiently fitted when the number of latent variables is high. The integrated nested Laplacian approximation (INLA), an alternative method to Monte Carlo, permits inference for even large spatio-temporal models; and the *SPDE approach* allows us to specify models, for which the correlation function might not even have a closed form. In this thesis, we show how recent advances can be combined to extend 2D + time models into the third spatial dimension.

Contents

Chapter 1

Introduction

Non-Bayesian approaches dominated statistical theory and practice for most of the last century, but the last few decades have seen a re-emergence of Bayesian methods. This has been driven more by the availability of new computational techniques than by what many would see as the theoretical and logical advantages of Bayesian thinking.
— Andrew Gelman et al. [10]

Integrated nested Laplacian approximation (INLA) [26] is a statistical framework for performing fast, approximate Bayesian inference, providing a performant alternative to the Markov chain Monte Carlo method (MCMC). It has become particularly popular in the (geo-)spatial and spatio-temporal modelling communities, and has been used to model problems in a wide variety of fields, such as:

- in atmospheric science, to model precipitation [13];
- in ecology, to model seal encounters [6] and shark bycatch [7];
- in health science, to model malaria spread [23] and analyse MRI data [29];
- in environmental science, to model particulate matter pollution [3].

For a comprehensive review, see [17].

Until now, the `R-INLA` package has only supported one to two space dimensions, all the while our reality is three-dimensional. Recent developments add preliminary support for spatial modelling in three dimensions [16], as well as non-separable space-time models [19] to INLA—we aim to demonstrate that they can be effectively combined.

We start by giving an brief summary of Bayesian inference and spatio-temporal modelling using INLA in chapter 2—this should by no means be comprehensive introduction to the subject, and we refer the reader to more in-depth texts when the need arises. In chapter 3, we describe our process and validation setup; while in chapter 4 we delve into the details of how 3D + time models can be codes in `R-INLA`.

1.1. Acknowledgements

The author would like to thank cand. PhD. Lisa Gaedke-Merzäuser, for her patience, encouragement, and all-round help; Dr. Elias T. Krainski for timely replies and changes to [14]; and Dr. Roger Käppeli, for introducing the author to XDMF [35]; furthermore, the author's thanks go out to numerous friends, family, and acquaintances, for their everlasting support.

Chapter 2

Background

2.1. Bayesian inference

Supposing you have some data, how should you use it to learn about the world? There is no uniquely correct answer to this question. Lots of approaches, both formal and heuristic, can be effective. But one of the most effective and general answers is to use Bayesian data analysis. Bayesian data analysis takes a question in the form of a model and uses logic to produce an answer in the form of probability distributions.
— Richard McElreath [21]

In the Bayesian framework, all model parameters $\boldsymbol{\theta}$ are treated as random variables. The aim is then to calculate an estimate of the *joint posterior probability* of the parameters given the data, $\mathbb{P}(\boldsymbol{\theta} \mid \boldsymbol{y})$. This is done with help of Bayes' theorem,

$$\mathbb{P}(\boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{\mathbb{P}(\boldsymbol{y} \mid \boldsymbol{\theta})\, \mathbb{P}(\boldsymbol{\theta})}{\mathbb{P}(\boldsymbol{y})} \propto \mathbb{P}(\boldsymbol{y} \mid \boldsymbol{\theta})\, \mathbb{P}(\boldsymbol{\theta}), \tag{2.1}$$

wherein the *marginal likelihood* of the data, $\mathbb{P}(\boldsymbol{y})$, is difficult to compute. Furthermore, we rarely require full knowledge about the distribution $\mathbb{P}(\boldsymbol{\theta} \mid \boldsymbol{y})$—instead, usual questions one seeks answers to, are:

- Which model parameters $\boldsymbol{\theta}$ explain the data $\boldsymbol{y}$ best? This is the mode of $\mathbb{P}(\boldsymbol{\theta} \mid \boldsymbol{y})$, also called the *maximum a posteriori* estimate thereof.

- What is the distribution of a single parameter θ_i? This is characterised by a variety of summary statistics, such as the mean, median, mode, standard deviation, and quantiles; all of which require the *marginal posterior probability*,

$$\mathbb{P}(\theta_i \mid \boldsymbol{y}) = \int \mathbb{P}(\boldsymbol{\theta} \mid \boldsymbol{y}) \mathrm{d}\boldsymbol{\theta}_{-i}. \tag{2.2}$$

  We write $\boldsymbol{\theta}_{-i}$ to mean $\{\theta_j \in \boldsymbol{\theta} \mid i \neq j\}$.

To evaluate the aforementioned integrals, a number of methods are available. Conjugate priors can be used, when the distributions are simple—which is rarely true in practice. Grid methods, while simple to implement, do not scale well, and have mostly pedagogical value [21]. Today's methods of choice are Monte Carlo and variational Bayesian methods; these, however, become intractable when the number of parameters increases.

For an accessible introduction to Bayesian inference, we recommend reading [21], while an even more in-depth treaty thereof can be found in [10].

2.2. Integrated nested Laplacian approximation

The key operation in Bayesian inference is to compute high-dimensional integrals. An old approximate technique is the Laplace method or approxi- mation, which dates back to Pierre-Simon Laplace (1774). This simple idea approximates the integrand with a second-order Taylor expansion around the mode and computes the integral analytically. By developing a nested version of this classical idea, combined with modern numerical techniques for sparse matrices, we obtain the approach of integrated nested Laplace approximations (INLA) to do approximate Bayesian inference for latent Gaussian models (LGMs).
— Håvard Rue et al. [27]

Integrated nested Laplacian approximation (INLA) [26] is an alternative to Monte Carlo and variational Bayesian methods. It can be used to estimate posterior marginal distributions, and does this in an efficient manner—by approximating an underlying Gaussian random field (GRF) with a Gaussian Markov random field (GMRF), it can use highly optimised sparse-matrix libraries to do the estimation [9]. Additionally, the computations it executes are easily parallelisable. This makes it well suited for models with many latent variables, which often arise in spatial and spatio-temporal modelling.

Instead of explaining in detail all the constituent parts and approximations of INLA here, we point the reader to [27] for an accessible explanation thereof, as well as the original publication [26]. Information regarding using INLA can be found for example in [12], or specifically for spatial modelling, in [15].

2.3. Spatial modelling

Use the Matérn model.
— Michael L. Stein [30]

A core tenet of spatial modelling is that things that are closer in space are more strongly correlated. The choice of correlation function[1] is therefore quite important. The arguably most common choice is the Matérn correlation function,

$$\mathrm{cor}_{\gamma_s,\nu_s}(\boldsymbol{x},\boldsymbol{y}) = \frac{(\gamma_s \cdot \|\boldsymbol{x}-\boldsymbol{y}\|)^{\nu_s} \cdot \mathrm{K}_{\nu_s}(\gamma_s \cdot \|\boldsymbol{x}-\boldsymbol{y}\|)}{2^{(\nu_s-1)} \cdot \Gamma(\nu_s)}, \tag{2.3}$$

where $\|\boldsymbol{x}-\boldsymbol{y}\|$ is the Euclidean distance between points $\boldsymbol{x}$ and $\boldsymbol{y}$; $\Gamma(z)$ is the Gamma function; $\mathrm{K}_\alpha(z)$ is the modified Bessel function of the second kind of order α; and

$$\gamma_s = \frac{\sqrt{8\nu_s}}{\rho_s}. \tag{2.4}$$

[1]A correlation function is a measure for linear dependence between two variables (here, two points of a random field), taking values in the range $[-1, 1]$. A correlation of 1 is indicative of a positive relationship ("as x increases, so does y"), and a value of -1 of an inverse relationship ("as x increases, y decreases"). Note, that a linear relationship is assumed; if the variables are related in some other way, correlation will not show that.

Discovered by Bertil Matérn [20], and popularised by Michael L. Stein [30], it has a number of appealing properties: it allows for control of the smoothness via the parameter $\nu_s > 0$, as the underlying Gaussian process is $(\lceil \nu_s \rceil - 1)$-times differentiable in the mean-square sense; it is a generalisation of the exponential ($\nu_s = 1/2$), the second- ($\nu_s = 3/2$) and third-order ($\nu_s = 5/2$) autoregressive correlation functions, as well as of the Gaussian radial basis function ($\lim_{\nu_s \to \infty}$). The range parameter $\rho_s > 0$ controls the distance of the correlation, with $\mathrm{cor}(\boldsymbol{x}, \boldsymbol{y}) \approx 0.14$ when $\|\boldsymbol{x} - \boldsymbol{y}\| = \rho_s$.

Multiplying the Matérn correlation function with the variance of the Gaussian field σ_e^2, we get the Matérn covariance function,

$$\mathrm{cov}_{\gamma_s, \nu_s, \sigma_e}(\boldsymbol{x}, \boldsymbol{y}) = \sigma_e^2 \cdot \mathrm{cor}_{\gamma_s, \nu_s}(\boldsymbol{x}, \boldsymbol{y}) = \sigma_e^2 \cdot \frac{(\gamma_s \cdot \|\boldsymbol{x} - \boldsymbol{y}\|)^{\nu_s} \cdot \mathrm{K}_{\nu_s}(\gamma_s \cdot \|\boldsymbol{x} - \boldsymbol{y}\|)}{2^{(\nu_s - 1)} \cdot \Gamma(\nu_s)}. \tag{2.5}$$

An important result of [34] is that a Gaussian field $\boldsymbol{u}$ with Matérn covariance $\mathrm{cov}_{\gamma_s, \nu_s, \sigma_e}(\boldsymbol{x}, \boldsymbol{y})$ solves the following stochastic PDE (SPDE)

$$\begin{cases} \left(\gamma_s^2 - \Delta\right)^{\frac{\alpha_s}{2}} \boldsymbol{u}(\boldsymbol{x}) = \mathcal{W}(\boldsymbol{x}) & \text{on } \Omega_s \\ \dfrac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} = 0 & \text{on } \partial\Omega_s \end{cases} \tag{2.6}$$

with Δ being the Lapalacian, $\mathcal{W}$ spatial white noise, and

$$\alpha_s = \nu_s + \frac{d}{2} > 0 \tag{2.7}$$

where d is the spatial dimension. This equation is well-studied for positive integer values of $\alpha_s \in \mathbb{Z}_{>0}$, although extensions to arbitrary $\alpha_s \in \mathbb{R}_{>0}$ have been attempted [18]. Of particular note we find the case where $\alpha_s = 2$, as then the form of eq. (2.6) is the stochastic version of the screened Poisson equation,

$$(\Delta - \lambda^2)\boldsymbol{u}(\boldsymbol{x}) = -f(\boldsymbol{x}), \tag{2.8}$$

which is used in physics to model the screening effect in e.g. plasmas. A charged particle will repel other particles of the same charge, leading to a region around the itself where the charge density is lower. At larger distances, this "screening hole" behaves like yet another particle, with opposite charge. The net effect is, that, at large distances, the Coulomb force of particle and its "screening hole" cancel each other out, such that only short-range interactions are of importance. This mirrors the general behaviour of the Matérn covariance.

Defining a pseudo-differential operator $\mathcal{L}_s := (\gamma_s^2 - \Delta)^{\frac{\alpha_s}{2}}$, eq. (2.6) can be used to construct a discrete version of the precision operator $\mathcal{Q}_s = \mathcal{L}_s \mathcal{L}_s$, the precision matrix $\boldsymbol{Q}_{s, \alpha_s, \gamma_s}$ in the so called *SPDE approach*[2] [18]: $\boldsymbol{Q}_{s, \alpha_s, \gamma_s}$ can be expressed in terms of the

[2]Being the inverse of the covariance matrix $\boldsymbol{\Sigma}$, the precision matrix must be symmetric positive-definite (spd). Construction of spd matrices is not a trivial task; the advantage of the *SPDE approach* lies in making this problem tractable.

mass matrix $\boldsymbol{C}_s$ and stiffness matrix $\boldsymbol{G}_s$ when eq. (2.6) is discretised using the finite element method (FEM).

$$\boldsymbol{C}_s = (c_{i,j}) \qquad c_{i,j} = \langle \psi_i, \psi_j \rangle \tag{2.9}$$

$$\boldsymbol{G}_s = (g_{i,j}) \qquad g_{i,j} = \langle \nabla\psi_i, \nabla\psi_j \rangle \tag{2.10}$$

Define matrix $\boldsymbol{K}_{s,\gamma_s}$ to be

$$\boldsymbol{K}_{s,\gamma_s} := \gamma_s^2 \boldsymbol{C}_s + \boldsymbol{G}_s \tag{2.11}$$

Then, for integer α_s, the precision matrix $\boldsymbol{Q}_{s,\alpha_s,\gamma_s}$ can be expressed as follows:

$$\boldsymbol{Q}_{s,\alpha_s,\gamma_s} = \boldsymbol{C}_s^{\frac{1}{2}} \left(\boldsymbol{C}_s^{-\frac{1}{2}} \boldsymbol{K}_{s,\gamma_s} \boldsymbol{C}_s^{-\frac{1}{2}} \right)^{\alpha_s} \boldsymbol{C}_s^{\frac{1}{2}} = \begin{cases} \boldsymbol{K}_{s,\gamma_s} & \text{iff } \alpha_s = 1 \\ \boldsymbol{K}_{s,\gamma_s} \boldsymbol{C}_s^{-1} \boldsymbol{K}_{s,\gamma_s} & \text{iff } \alpha_s = 2 \\ \boldsymbol{K}_{s,\gamma_s} \boldsymbol{C}_s^{-1} \boldsymbol{Q}_{s,\alpha_s-2,\gamma_s} \boldsymbol{C}_s^{-1} \boldsymbol{K}_{s,\gamma_s} & \text{iff } \alpha_s > 2. \end{cases} \tag{2.12}$$

Both the mass matrix $\boldsymbol{C}_s$ and the stiffness matrix $\boldsymbol{G}_s$ are sparse due to the nature of FEM; this cannot be said of the matrix $\boldsymbol{C}_s^{-1}$, as it will experience fill-in. The consistent mass matrix $\boldsymbol{C}_s$ can however be approximated by a lumped mass matrix $\tilde{\boldsymbol{C}}_s$, whose inverse is also sparse. In [18], the authors suggest using the diagonal matrix

$$\tilde{\boldsymbol{C}}_s = (\tilde{c}_{i,j}) \qquad \tilde{c}_{i,j} := \begin{cases} \langle \psi_i, 1 \rangle & \text{iff } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{2.13}$$

Substituting $\boldsymbol{C}_s$ for $\tilde{\boldsymbol{C}}_s$ in eq. (2.12) gives us the sparse version of the precision matrix

$$\tilde{\boldsymbol{Q}}_{s,\alpha_s,\gamma_s} = \tilde{\boldsymbol{C}}_s^{\frac{1}{2}} \left(\tilde{\boldsymbol{C}}_s^{-\frac{1}{2}} \boldsymbol{K}_{s,\gamma_s} \tilde{\boldsymbol{C}}_s^{-\frac{1}{2}} \right)^{\alpha_s} \tilde{\boldsymbol{C}}_s^{\frac{1}{2}} = \begin{cases} \boldsymbol{K}_{s,\gamma_s} & \text{iff } \alpha_s = 1 \\ \boldsymbol{K}_{s,\gamma_s} \tilde{\boldsymbol{C}}_s^{-1} \boldsymbol{K}_{s,\gamma_s} & \text{iff } \alpha_s = 2 \\ \boldsymbol{K}_{s,\gamma_s} \tilde{\boldsymbol{C}}_s^{-1} \tilde{\boldsymbol{Q}}_{s,\alpha_s-2,\gamma_s} \tilde{\boldsymbol{C}}_s^{-1} \boldsymbol{K}_{s,\gamma_s} & \text{iff } \alpha_s > 2. \end{cases} \tag{2.14}$$

Note, that while it is possible to also substitute $\boldsymbol{C}_s$ for $\tilde{\boldsymbol{C}}_s$ in $\boldsymbol{K}_{s,\gamma_s}$, this is not necessary to get the desired sparsity.

A detailed derivation of the above procedure can be found in [18], an accessible and intuitive summary in [22], and its use in INLA in [15]. Furthermore, the SPDE approach gives us a means to specify models and their precision matrices, for which writing down the covariance function might be difficult or not possible. This can be done by altering eq. (2.6) in a physically motivated manner—[18] provides examples for non-stationary and non-isotropic models. This technique is also used in [19] to specify a non-separable space-time model, as described in section 2.4.

2.4. Spatio-temporal modelling

> [...] *we note that these operators are almost never encountered when modeling physical reality, hence, separable models are typically not physically motivated models for the spatio-temporal process.*
> — Finn Lindgren et al. [19]

The simplest spatio-temporal model can be constructed by using an SPDE model for the spatial domain, and an autoregressive model AR(1) for the temporal domain. In this case, the precision matrix $\boldsymbol{Q}_{st}$ can be constructed as the Kronecker product of the spatial and temporal precision matrices of the respective parts,

$$\boldsymbol{Q}_{st} = \boldsymbol{Q}_s \otimes \boldsymbol{Q}_t. \tag{2.15}$$

Since the model is separable, it is equivalent to fitting a spatial model for each time-step, which, as noted in [19], is "not physically motivated".

Instead, a non-separable model can be specified by modifying eq. (2.6) as outlined in section 2.3. There are many ways do achieve this goal— an overview, as well as a discussion and critique, of such modifications is given in [19], and a new class of models, called *diffusion-based extension of the Gaussian Matérn field* (*DEMF*), is proposed. This is based on the SPDE

$$\begin{cases} \left(-\gamma_t^2 \dfrac{\partial^2}{\partial t^2} + \left(\left(\gamma_s^2 - \Delta\right)^{\frac{\alpha_s}{2}}\right)^2\right)^{\frac{\alpha_t}{2}} \boldsymbol{u}(\boldsymbol{x}, t) = \mathrm{d}\mathcal{E}(\boldsymbol{x}, t) & \text{on } \Omega \\ \dfrac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} = 0 & \text{on } \partial\Omega, \end{cases} \tag{2.16}$$

with $\mathcal{E}(\boldsymbol{x}, t)$ denoting Gaussian noise that is correlated in space and white in time. In the case of $\gamma_t = 0$ and $\alpha_t = 1$, this reduces to eq. (2.6). The DEMF family is characterised by three smoothness parameters α_t, α_s and α_e, which define the behaviour and separability of the model. Following the notation of [19], we write DEMF$(\alpha_t, \alpha_s, \alpha_e)$ when discussing a specific model where these parameters are fixed. Additionally, three scale parameters γ_t, γ_s and γ_e need to be specified. We refer to these as *SPDE parameters*, since they have no physical meaning—which is why we usually work with the *interpretable parameters*,

$$\sigma_e = \sqrt{\frac{C_{1,\alpha_t} \cdot C_{d,\alpha}}{\gamma_t \gamma_e^2 \gamma_s^{(2\alpha - d)}}} \tag{2.17}$$

$$\rho_s = \frac{\sqrt{8\nu_s}}{\gamma_s} \tag{2.18}$$

$$\rho_t = \frac{\gamma_t \sqrt{8\left(\alpha_t - \frac{1}{2}\right)}}{\gamma_s^{\alpha_s}}, \tag{2.19}$$

where

$$C_{d,\alpha} := \frac{\Gamma\left(\alpha - \frac{d}{2}\right)}{(4\pi)^{\frac{d}{2}} \cdot \Gamma(\alpha)} \tag{2.20}$$

is a scaling constant,

$$\alpha := \alpha_e + \alpha_s\left(\alpha_t - \frac{1}{2}\right) > \frac{d}{2} \tag{2.21}$$

is the model order, and

$$\nu_s = \alpha - \frac{d}{2} \tag{2.22}$$

$$\nu_t = \min\left\{\alpha_t - \frac{1}{2}, \frac{\nu_s}{\alpha_s}\right\} \tag{2.23}$$

are the spatial and temporal smoothness parameters, respectively. We note that eq. (2.18) and eq. (2.4) are equivalent—the interpretation of the spatial smoothness ν_s is the same as in the purely spatial model for a fixed time t; its value, however, now depends on not only α_s, but also α_t and α_e (compare eqs. (2.7) and (2.22)).

Following a procedure similar to the one in section 2.3, we can use the *SPDE approach* to write down an expression approximating the precision matrix

$$\boldsymbol{Q}_{st} \approx \tilde{\boldsymbol{Q}}_{st} = \gamma_e^2 \sum_{k=0}^{2\alpha_t} \gamma_t^k \boldsymbol{Q}_{t,\alpha_t,\frac{k}{2}} \otimes \tilde{\boldsymbol{Q}}_{s,\alpha_e+\alpha_s(\alpha_t-k/2),\gamma_s}, \tag{2.24}$$

with $\boldsymbol{Q}_{s,\alpha_s,\gamma_s}$ as in eq. (2.14). The expression for $\boldsymbol{Q}_{t,\alpha_t,\frac{k}{2}} \in \mathbb{R}^{N\times N}$ is more involved; we therefore limit ourselves to the case $\alpha_t = 1$ where

$$\boldsymbol{Q}_{t,1,0} = \boldsymbol{C}_t = (c_{i,j}) \qquad c_{i,j} = \langle \phi_i, \phi_j \rangle \tag{2.25}$$

$$\boldsymbol{Q}_{t,1,1/2} = (q_{i,j}) \qquad q_{i,j} = \begin{cases} 1/2 & \text{iff } (i,j) \in \{(1,1),(N,N)\} \\ 0 & \text{otherwise} \end{cases} \tag{2.26}$$

$$\boldsymbol{Q}_{t,1,1} = \boldsymbol{G}_t = (g_{i,j}) \qquad g_{i,j} = \langle \nabla\phi_i, \nabla\phi_j \rangle, \tag{2.27}$$

with $\boldsymbol{C}_t$ and $\boldsymbol{G}_t$ being the temporal mass matrix and the temporal stiffness matrix from the FEM discretisation, respectively; and $\boldsymbol{Q}_{t,1,1/2}$ a term which corrects for stationarity. We note that all of the above matrices are sparse due their construction. For a detailed derivation, discussion and proofs we divert the reader to [1, 19].

Chapter 3

Extending **R-INLA** for 3D + time

3.1. Meshing

The first obstacle in extending INLA [26] to three spatial dimensions and time, albeit not a particularly tall one, was to generate appropriate discretisations of the domain. For reasons of simplicity, we decided to consider the spatial discretisation constant throughout the time domain; therefore, spatial and temporal meshes could be generated independently, and later combined using the Cartesian graph product.

Definition 1 (Cartesian graph product) The Cartesian graph product $G \square H$ of graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ is the graph $(V_G \times V_H, \{((u_G, u_H), (v_G, v_H)) \mid (u_G = v_G \wedge u_H \sim v_H) \vee (u_G \sim v_G \wedge u_H = v_H)\})$.

An example can be seen in fig. 3.1.

As the temporal mesh is one-dimensional, creating it reduces to specifying the distances between neighbouring time knots; the appropriate functionality is already provided by INLA via the `inla.mesh.1d` function.

The mesher that is bundled with INLA, `fmesher`, only supports generating two-dimensional meshes. Instead of extending it to support three-dimensional meshing, we decided to use a well established, off-the-shelf mesher. To this end, we integrated both `TetGen` and `Gmsh` with R and INLA.

`TetGen` [28] is a freely-accessible programme for generating three-dimensional meshes using constrained Delaunay tetrahedralisation; given a set of points including their coordinates, as well as a domain boundary, it creates an unstructured mesh whose elements

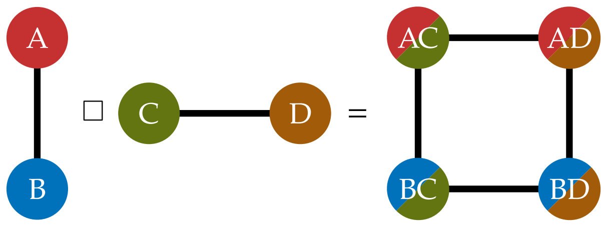


Figure 3.1.: Example of the Cartesian graph product of K_2 and K_2.

are tetrahedra fullfilling certain quality criteria[1] (constrained *Delaunay tetrahedralisation*), all the while preserving the geometry of the boundary (*constrained* Delaunay tetrahedralisation). It is written in C++ and provides a command-line interface (CLI) for usage as well as a C++ application programming interface (API) for integration with other applications. We use this API to provide a set of bindings to R, giving the resulting package the name `tetgenr` [32].

Since a boundary definition needs to be provided by the user even for convex domains, we optionally require the R package `geometry` [31] in `tetgenr`; this allows us to automate the definition of the boundary for convex domains by computing the convex hull using `geometry::qhull( ... )`.

`Gmsh` [11] is an alternative to `TetGen`, upon which it builds and expands to provide general meshing facilities for two- and three-dimensional domains. In addition to unstructured triangular (in 2D) and tetrahedral (in 3D) grids, it can also use quads or hexahedra as elements. Furthermore, the programme comes with a computer-aided design (CAD) module for manipulating geometries. Similarly to `TetGen`, `Gmsh` provides a user-facing CLI as well as a C++ API. Due to the increased complexity of the programme, the usage of both of these is more involved than with `TetGen`—whereas in `TetGen` the user provides a set of points, their coordinates, and a set of boundary triangles, in `Gmsh` the boundaries need to be assembled from lower dimensional elements; that is to say that a triangle has to be specified in terms of its edges, which in turn are line segments defined by their endpoints. `Gmsh` additionally provides many more options to tweak the meshing process than `TetGen`. Just as in the case of `TetGen`, we provide an R wrapper around this API, which we call `gmshr` [32].

A comparison of the meshes generated by `TetGen` and `Gmsh` was outside of the scope of this work, both in terms of their quality, as well of their runtime. That being said, both programmes are well established and provided satisfactory results. We do think that `Gmsh` is the more versatile software, having more options and parameters that can be set. We also point out that the time and effort of generating a mesh is dwarfed by the computational effort of doing the model fitting itself.

3.2. Extending INLA to support 3D + time

The second obstacle, and the biggest one, in extending INLA to three spatial dimensions and time, was specifying the model in a way which it could be input into the INLA programme. This turned out to be more challenging than initially anticipated.

INLA does not support three-dimensional spatial models out of the box—this functionality is implemented in an experimental package called `inlamesh3d` [16]. Being

[1]`TetGen` [28] guarantees that elements have

- a lower bound on their dihedral angle
- an upper bound on their volume
- an upper bound on the ratio between their longest edge and the radius of the corresponding circumsphere

experimental, not all functionality which is available for one- and two-dimensional models in INLA is implemented in `inlamesh3d` [16]: for example, `inlamesh3d::inla.mesh3d.fem` does not implement the `order` parameter to output higher-order FEM matrices, instead failing silently without outputting an error message.

Similarly, non-separable spatio-temporal model support has not yet been upstreamed into INLA, and is being actively developed in the package `INLAspacetime` [14]. The initial version which we were working with was from September 2022 (commit 03a9421), and assumed the domain to be two-dimensional; we had access to another working implementation from another project, which used a spherical surface as the spatial domain, ie. a 2-manifold. We therefore set out to implement our own three-dimensional version based on [1]; however we found the description of the procedure in that paper to be incomplete. Having tried everything we could think of, we contacted the authors, who made us aware that [1] had been superseded by [19] in the meantime. In collaboration with them, we were able to contribute towards adding support for three-dimensional non-separable models to `INLAspacetime` (cf. chapter 4).

3.3. Visualisation

We wanted to be able to inspect the approximated posterior distribution, as computed by INLA, visually. While R and INLA support plotting of 2D fields very well, visualisation of 3D plots is somewhat limited. Firstly, when viewing 3D plots, some degree of interactivity is often preferable. While the `rgl` [24] package does implement some of these capabilities, we wanted offer well known tools. To this end, we attempted integration of INLA output with ParaView [25], a well-known visualisation programme.

We accomplish this by converting INLA output to extensible data model and format (XDMF), and visualising it in ParaView. XDMF [35] is a extensible markup language (XML)-based data interchange format for mesh-based plotting. It allows the user to define different meshes in terms of their geometry and topology, as well as scalar, vector and matrix values at nodes or cells. The time dimension is also supported, both for the mesh structure as well as the field values. An example of how such a file might look like is shown in listing 1. Data can either be stored inline in the same file as text, or out-of-line as datasets in a separate hierarchical data format, version 5 (HDF5) [33] file.

Initially, we planned to represent nodal and cell values as arrays of dimension $(d + 1)$ in XDMF, with the added dimension being time. However, we discovered that support for XDMF in ParaView is incomplete, and array slicing (called *hyperslabs* in XDMF) is not supported [36]. Therefore, we save values for each time-step in their own HDF5 dataset—this has the nice side-effect of making us support meshes which change with time in the future.

```xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Xdmf Version="2.0">
3    <Domain>
4      <Grid Name="mesh" GridType="Collection" CollectionType="Temporal">
5        <Grid Name="mesh 0" GridType="Uniform">
6          <Time TimeType="single" Value="0"/>
7          <Geometry Type="XYZ">
8            <DataItem Format="HDF" Dimensions="4 3" NumberType="Float">result.h5:/mesh/geometry</DataItem>
9          </Geometry>
10         <Topology TopologyType="Tetrahedron">
11           <DataItem Format="HDF" Dimensions="1 4" NumberType="Int">result.h5:/mesh/topology</DataItem>
12         </Topology>
13         <Attribute Name="value" AttributeType="Scalar" Center="Node">
14           <DataItem Format="HDF" Dimensions="4" NumberType="Float">result.h5:/mesh/0/node_attributes/value</DataItem>
15         </Attribute>
16       </Grid>
17       <Grid Name="mesh 1" GridType="Uniform">
18         <Time TimeType="single" Value="1"/>
19         <Geometry Type="XYZ">
20           <DataItem Format="HDF" Dimensions="4 3" NumberType="Float">result.h5:/mesh/geometry</DataItem>
21         </Geometry>
22         <Topology TopologyType="Tetrahedron">
23           <DataItem Format="HDF" Dimensions="1 4" NumberType="Int">result.h5:/mesh/topology</DataItem>
24         </Topology>
25         <Attribute Name="value" AttributeType="Scalar" Center="Node">
26           <DataItem Format="HDF" Dimensions="4" NumberType="Float">result.h5:/mesh/1/node_attributes/value</DataItem>
27         </Attribute>
28       </Grid>
29     </Grid>
30   </Domain>
31 </Xdmf>
```

Listing 1: A simple XDMF file with two time-points (lines 3–16 and 17–28), a mesh consisting of 4 points (lines 7–9 and 19–21) defining a single tetrahedral cell (lines 10–12 and 22–24) and scalar nodal values (lines 13–15 and 25–27). The mesh does not change between the time-steps, as evidenced by the **`<Geometry>`** and **`<Topology>`** elements referencing the exact same data (`result.h5:/mesh/geometry` and `result.h5:/mesh/topology` respectively).

`Mesh.MeshSizeMax`	Node count	Element count
0.2	367	1196
0.15	471	1631
0.125	763	2917
0.11	1049	4160

Table 3.1.: Summary of generated spatial meshes.

3.4. Synthetic data generation

We started with the simplest generative model we could conceive: noisy observation of constant data ($\equiv 0$) on a unit domain $\Omega_{st} = \Omega_s \times \Omega_t$.

$$\begin{aligned} \boldsymbol{y} &\sim \boldsymbol{u}(\boldsymbol{x}, t) + \mathcal{N}(0, \sigma_n^2) && (3.1)\\ \boldsymbol{u}(\boldsymbol{x}, t) &\sim \mathcal{N}(0, \boldsymbol{Q}_{st}^{-1}) && (3.2)\\ \boldsymbol{x} &\in \Omega_s = [0,1]^3 \subseteq \mathbb{R}^3 && (3.3)\\ t &\in \Omega_t = [0,1] \subseteq \mathbb{R} && (3.4) \end{aligned}$$

A spatio-temporal DEMF(1, 2, 1) model was chosen, meaning $\alpha_e := 1$, $\alpha_s := 2$ and $\alpha_t := 1$.

By specifying all model parameters, we can generate synthetic data by sampling from the corresponding distributions, and doing any necessary transformations. In our example, we set the *interpretable parameters*

$$\begin{cases} \rho_t := 0.3 & \rho_s := 0.5 \\ \sigma_e := 1 & \sigma_n := 0.1 \end{cases} \tag{3.5}$$

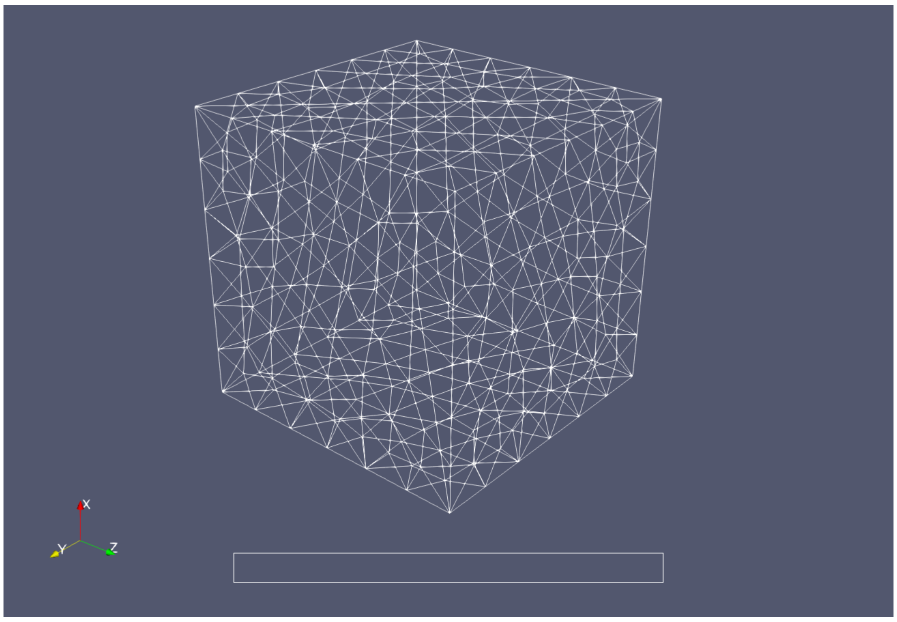
and derive the *SPDE parameters* using eqs. (2.17) to (2.23). We discretise the spatial domain Ω_s using `Gmsh` [11] with different values of `Mesh.MeshSizeMax`, in order to get meshes of different resolution; the statistics of the resulting meshes can be seen in table 3.1; and a visualisation in fig. 3.2. For the time domain, we choose a uniform discretisation with $n_t := 5$ time-steps of length $\Delta t = 0.2$. Equation (2.24) becomes

$$\boldsymbol{Q}_{st} \approx \tilde{\boldsymbol{Q}}_{st} = \gamma_e^2 \left(\boldsymbol{Q}_{t,1,0} \otimes \tilde{\boldsymbol{Q}}_{s,3,\gamma_s} + 2\gamma_t \boldsymbol{Q}_{t,1,\frac{1}{2}} \otimes \tilde{\boldsymbol{Q}}_{s,2,\gamma_s} + \gamma_t^2 \boldsymbol{Q}_{t,1,1} \otimes \tilde{\boldsymbol{Q}}_{s,1,\gamma_s} \right), \tag{3.6}$$

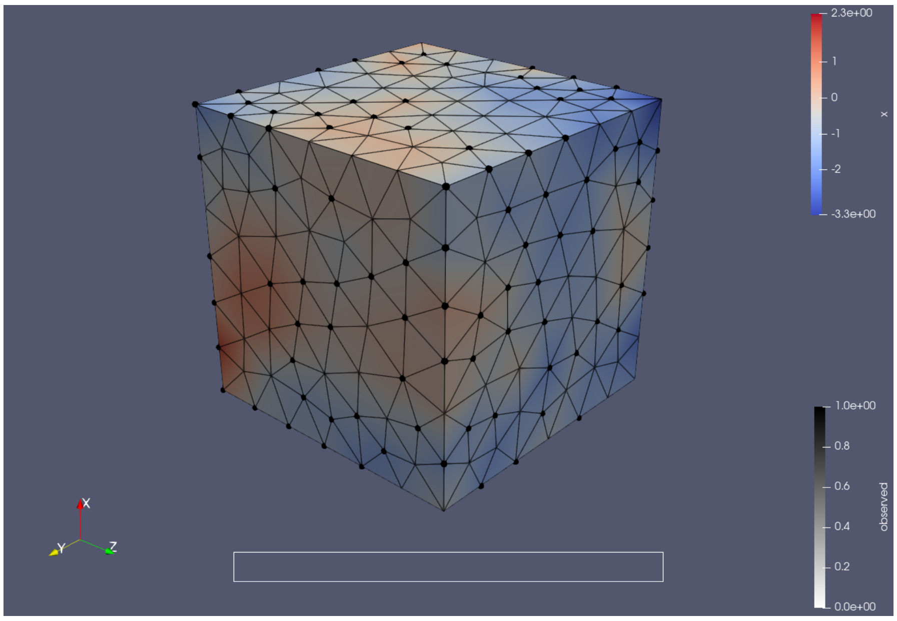
as we are using the DEMF(1, 2, 1) model (cf. section 2.4). We plot the sparsity pattern of $\tilde{\boldsymbol{Q}}_{st}$ in fig. 3.3. Having the precision matrix allow us to sample from $\mathcal{N}(0, \tilde{\boldsymbol{Q}}_{st}) + \mathcal{N}(0, \sigma_n^2)$ to generate our noisy sample data $\boldsymbol{y}$. Note that the zero mean property of the field $\boldsymbol{u}(\boldsymbol{x}, t)$ and the noise $\mathcal{N}(0, \sigma_n^2)$ can trivially guaranteed by centering the samples.

3.5. Model fitting and parameter recovery

We split the generated dataset into two disjoint sets: a training dataset $\boldsymbol{y}_{\text{train}}$ consisting of β percent of the generated data, and a testing dataset $\boldsymbol{y}_{\text{test}}$ with the rest of the data.



(a) The spatial mesh corresponding to `Mesh.MeshSizeMax` = 0.2, with 367 nodes and 1196 tetrahedral elements.



(b) A generated dataset for time-step $t = 0$.

Figure 3.2.: Visualisation of the generated dataset $\boldsymbol{y}$.

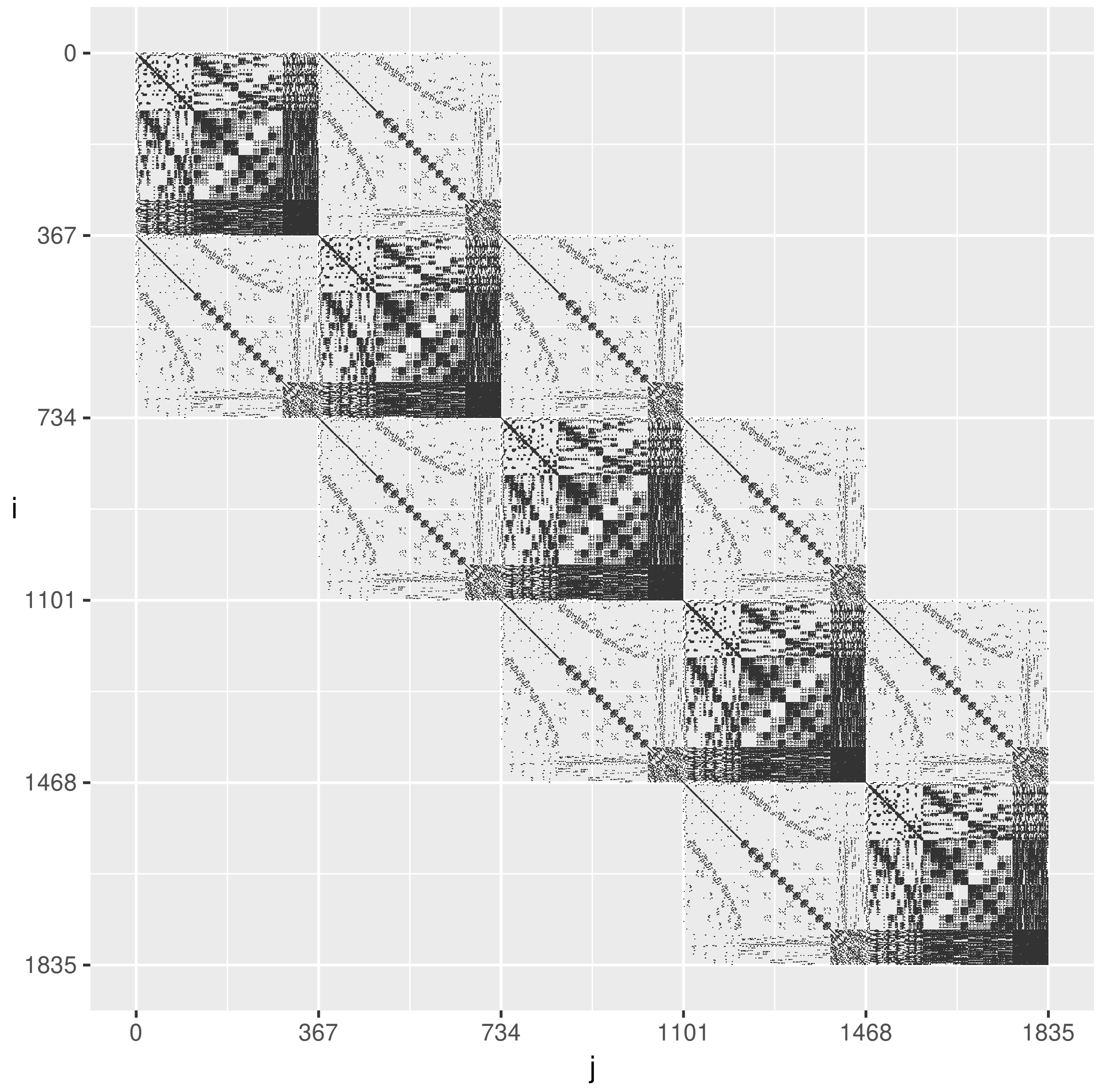
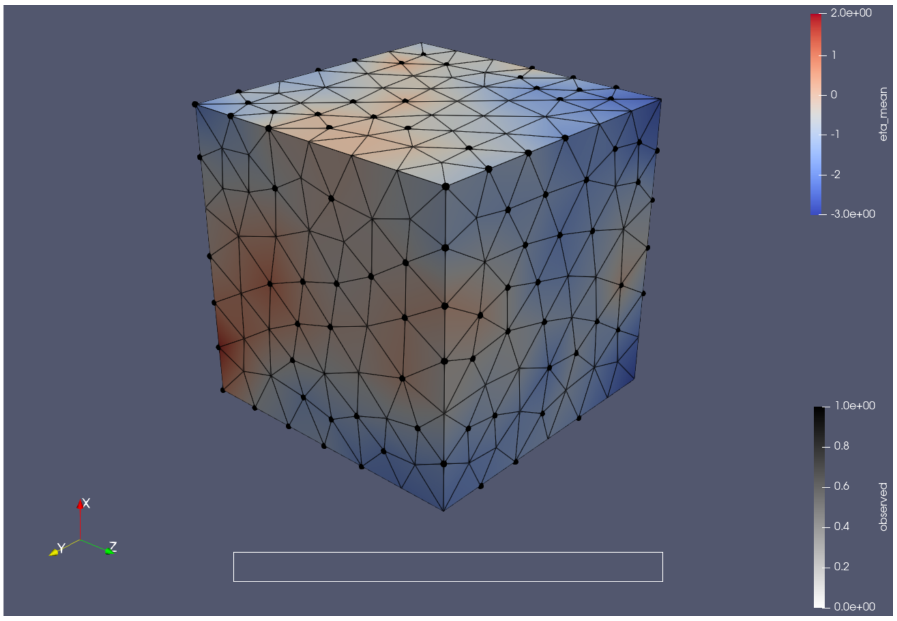


Figure 3.3.: The sparsity pattern of $\tilde{Q}_{st} = (\tilde{q}_{i,j})$. Recall that we have 367 spatial nodes and 5 time-steps in our mesh. The spatial mesh is clearly visible as blocks on the diagonal, replicated once for each time-steps. Due to the non-separable nature of the model, neighbouring time-steps are interconnected (compare eqs. (2.15) and (3.6)).

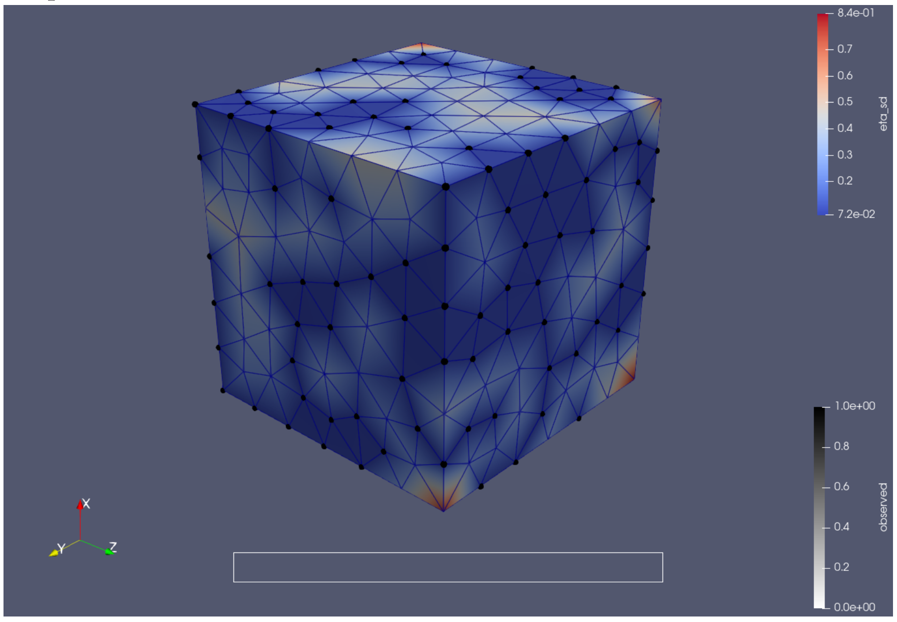
While we choose to perform the split only in space and not in time (i.e. data for a spatial point is either available for all time-steps, or for none of them)—in fact, not only can the availability of data vary both throughout space and time, but the observations need not even coincide with the mesh nodes, as they are interpolated unto them using the projector matrix $\boldsymbol{A} \in \mathbb{R}^{N_{\text{data}} \times N_{\text{mesh}}}$. It can be constructed by transforming the global Cartesian coordinates of each data point into local coordinates of the element in which that data point resides—linearly interpolating in the time dimension and using barycentric coordinates for the spatial dimension. Since in our experiments data points always coincide with the mesh nodes, $\boldsymbol{A}$ takes the form of a unit matrix, with rows corresponding to unobserved points removed. Lastly, we specify priors on the model parameters ρ_t, ρ_s, σ_e and σ_n, using penalised complexity priors [8], where the parameter value used to synthesise the data is the median of the prior. We then invert the generating procedure, and attempt to fit a DEMF$(1, 2, 1)$ model to the training dataset $\boldsymbol{y}_{\text{train}}$.

Having fitted the model to the synthetic data, we compare the estimated parameters ρ_t, ρ_s, σ_e and σ_n to the values we used for model generation—and consider them successfully recovered if the generative value lies in the 95% confidence interval (2.5%–97.5%)[2]; following the procedure outlined in the `inlamesh3d` [16] vignette. Lastly, we visually inspect the model output by visualising it; which can be seen in fig. 3.4.

[2]Note that the underlying processes are stochastic, and it cannot be expected that we always recover the generating value—the sample from $\tilde{\boldsymbol{Q}}_{st}$ might be extreme an have parameter values that are different from those we use to generate $\tilde{\boldsymbol{Q}}_{st}$



(a) Mean of the posterior distribution, as predicted by INLA. Visual comparison with fig. 3.2b shows little difference, which we interpret as good performance of the model.



(b) Plotted in colour is the standard deviation of the data, which can be taken as a measure of uncertainty of the model. Note, that it is smallest, where training data are present (denoted by black spheres), and increases, as the distance to these observations increases. Additionally, note that the uncertainty is the largest on the corner nodes, where no data is available, even when surrounded by other observations. This is a boundary effect, which would require adding ghost boundary cells to remediate.

Figure 3.4.: Visualisation the model, fitted to our synthetic data, using the same mesh as in fig. 3.2a, at time-step $t = 0$.

Chapter 4

Software and Code

Throughout the course of this work, multiple new R packages were developed [32]: `tetgenr` and `gmshr` allow for interfacing with `TetGen` [28] and `Gmsh` [11]; `meshr` provides a unified interface to different meshers, and outputs data in XDMF [35]; `inla3dt` provides functionalities needed for 3D + time modelling in INLA, which are not present in `INLA`, `inlamesh3d` or `INLAspacetime`. The above packages are detailed in the following sections. In section 4.5, we provide some useful code snippets, which might be of use to those wishing to replicate or build upon our work—these were either deemed to small as to warrant packaging, or were not generalisable enough to make into their own functions.

Additionally, we were able to make some contributions to the package `INLAspacetime` [14]. These consist of commits `1335173`, `c598c49`, `4b07d37` and `6e3c924`, which make the package installable from source from within R, and, more importantly, add support for 3D + time models to that package.

4.1. `tetgenr`

The R package `tetgenr` is a thin wrapper around the `TetGen` [28] C API. The newest version of the `TetGen` source code is bundled with the package, and gets built together with it. `tetgenr` exports one function, `tetgenr::mesh( ... )`, which calls `TetGen`, and takes an $(N \times 3)$ matrix of N point coordinates as the first argument, `geometry`, which should be meshed. A $(M \times 3)$ matrix of boundary facets can be optionally provided in the second argument, `hull`, where each row are three 1-based indexes into `geometry`, which form a boundary triangle facet. The order of the indexes does not matter, as the orientation of the facets will be corrected by the function. If `hull` is not supplied, will default to calculating the convex hull using Qhull [2] by calling `geometry::convhulln( ... )`, if available; otherwise, an error will be thrown. Further arguments customisation of the meshing process, and map directly to `TetGen` options. The resulting mesh is returned as a named list, with `$geometry` being an $(N' \times 3)$ matrix of point coordinates (depending on the options, points from the input may be inserted, deleted, or merged, to make the mesh adhere to the desired quality); and `$topology` a $(M' \times 4)$ matrix of 1-based point indices of the tetrahedral cells, which constitute the mesh. An example of usage can be seen in listing 2. For further details, consult the package documentation [32].

```r
# define a unit cube
spatial_domain <- rbind(
    c(0, 0, 0),
    c(0, 0, 1),
    c(0, 1, 0),
    c(0, 1, 1),
    c(1, 0, 0),
    c(1, 0, 1),
    c(1, 1, 0),
    c(1, 1, 1)
)
# mesh it
spatial_mesh <- tetgenr::mesh(spatial_domain)
```

Listing 2: Example use of `tetgenr::mesh`.

```r
spatial_mesh <- gmshr::mesh(
    spatial_domain,
    Mesh.MeshSizeMax        = 0.15,
    Mesh.OptimizeThreshold = 0.3,
    Mesh.QualityType        = 2,
    Mesh.OptimizeNetgen     = 1,
    Mesh.RecombineAll       = 1,
    Mesh.Recombine3DAll     = 1
)
```

Listing 3: Example usage of `gmshr::mesh( ... )`, with some common meshing options set. `spatial_domain` is assumed to be the same as in listing 2.

4.2. `gmshr`

Similarly to `tetgenr`, the R package `gmshr` provides an R wrapper around the C++ API of `Gmsh` [11], which is compiled with the package, and does not need to be installed on the system. The function `gmshr::mesh( ... )` is called just like `tetgenr::mesh( ... )`, and returns the same list structure. The difference lies in the extra optional arguments—here, any option that is understood by `Gmsh` is understood and can be set—see section "7 Gmsh options", and, in particular, "7.4 Mesh options", of the `Gmsh` manual. An example call can be seen in listing 3, and further documentation is available with the package [32].

```r
spatial_domain <- rbind(
    c(0, 0, 0),
    c(0, 0, 1),
    c(0, 1, 0),
    c(1, 0, 0)
)
spatial_hull <- matrix(1L:4L, nrow = 1, ncol = 4)
# Mesh the same domain with three different backends
spatial_mesh_qhull <- meshr::qhull(spatial_domain,
  ↪ spatial_hull)
spatial_mesh_tetgen <- meshr::tetgen(spatial_domain,
  ↪ spatial_hull)
spatial_mesh_gmsh <- meshr::gmsh(spatial_domain, spatial_hull)
# convert any of them to INLA format
spatial_mesh <- inlamesh3d::inla.mesh3d(
    loc = spatial_mesh_gmsh$geometry,
    tv  = spatial_mesh_gmsh$topology
)
```

Listing 4: The package `meshr` allows for easy exchange of meshing backends. Once a mesh has been generated, it can easily be translated into an INLA-compatible datastructure.

4.3. `meshr`

The function of the R package `meshr` is two-fold: first, it provides a unified interface to the mesh-generating facilities of `geometry`, `tetgenr` and `gmshr`. Herewith, all three backends can be called with the same base arguments (`geometry` and optionally, `hull`) and return results in the same format (namely a list with members `$geometry` and `$topology`, as described in section 4.1). This allows for simple changing of the meshing backend, as can be seen in listing 4.

Second, `meshr` provides the function `meshr::write_xdmf( ... )` to output a mesh to XDMF [35] format, for visualisation in external software such as ParaView [25]. Both 2D and 3D meshes are supported; scalar and vector-valued attributes of nodes and cells can be specified through the arguments `node_attributes` and `cell_attributes`, respectively. These can also vary through time when passing a vector of time-steps via the argument `times`. Mesh geometry and topology is currently assumed to stay constant throughout time; this is only a limitation of the current implementation, and support could easily be implemented, if the need should arise. Further documentation can be found alongside the package [32].

```r
mesh <- list(
    geometry = rbind(
        c(0, 0, 0),
        c(0, 0, 1),
        c(0, 1, 0),
        c(1, 0, 0)
    ),
    topology = matrix(1L:4L, nrow = 1, ncol = 4)
)
meshr::write_xdmf(
    "mesh.h5",
    "mesh.xdmf",
    geometry = mesh$geometry,
    topology = mesh$topology,
    node_attributes = list(
        node_scalar = c(1, 2, 3, 4)
        node_vector = rbind(
            c(1, 0, 0),
            c(2, 0, 0),
            c(3, 0, 0),
            c(4, 0, 0)
        )
    ),
    cell_attributes = list(
        cell_scalar = c(-1),
        cell_vector = rbind(
            c(-1, 0, 0)
        )
    )
)
```

Listing 5: Example of outputting of a XDMF file, in three spatial dimensions, without time. `"mesh.h5"` and `"mesh.xdmf"` are the names of the files generated—the former is a HDF5 [33] file containing the data, the latter a XDMF [35] file describing the former. The names of the elements of `node_attributes` and `cell_attributes` specify what the attribute will be called in ParaView [25]. Values must be specified for each node or element; therefore, scalar attributes must have length N, and vector-valued attributes must be $(N \times d)$ matrices, where N is the number of nodes or elements, and d is the number of spatial dimensions (this is a limitation of the visualisation software, which can only display vectors of the same dimension as the mesh).

```r
meshr::write_xdmf(
    "mesh.h5",
    "mesh.xdmf",
    geometry = mesh$geometry,
    topology = mesh$topology,
    times = c(0, 1),
    node_attributes = list(
        node_scalar = cbind(
            c(1, 2, 3, 4),
            c(5, 6, 7, 8)
        )
        node_vector = array(1:24, c(4, 3, 2))
    ),
    cell_attributes = list(
        cell_scalar = cbind(c(-1), c(-2))
        cell_vector = array(-1:-6, c(1, 3, 2))
    )
)
```

Listing 6: Example analogous to listing 5, with two time-steps, $t = 0$ and $t = 1$. Attribute values must be specified for all n_t time-steps, and therefore have dimensions $(N \times n_t)$ and $(N \times d \times n_t)$, when scalar-valued or vector-valued, respectively.

```r
spatial_fem ← inla3dt::mesh2fem(spatial_mesh, order = 3L)
temporal_fem ← inla3dt::mesh2fem(temporal_mesh)
gamma ← inla3dt::interpretable2spde(
    sigma_e, rho_s, rho_t, d = 3,
    alpha = c(t = 1, s = 2, e = 1)
)$gamma
Q ← inla3dt::make_Q(spatial_fem, temporal_fem, gamma)
```

Listing 7: Example use of `inla3dt::make_Q( ... )`. Here, `spatial_mesh` and `temporal_mesh` are INLA [26] mesh objects.

4.4. `inla3dt`

The package `inla3dt` implements functionality needed for defining 3D + time models in INLA. The function `inla3dt::mesh2fem( ... )` generalises the functions `INLA::inla.mesh.2d.fem( ... )`, `INLA::inla.mesh.fem( ... )` and `inalmesh3d::inla.mesh3d.fem( ... )`; calling the correct one and extending support to arbitrary `orders`. For 1D meshes, matrices $\boldsymbol{M}_0 := \boldsymbol{Q}_{t,1,0}$, $\boldsymbol{M}_1 := \boldsymbol{Q}_{t,1,1/2}$ and $\boldsymbol{M}_2 := \boldsymbol{Q}_{t,1,1}$ are additionally returned as `$m0`, `$m1` and `$m2`, respectively (cf. eqs. (2.25) to (2.27)).

The function `inla3dt::make_model_121( ... )` defines a DEMF$(1, 2, 1)$ model. Depending on the argument `mode`, this is either the `cgeneric` model from the package `INLAspacetime` (`INLAspacetime::stModel.define( ... )`) or the `rgeneric` model from this package[1]. The corresponding precision matrix $\tilde{\boldsymbol{Q}}_{st}$ can be generated using `inla3dt::make_Q( ... )`[2], as demonstrated in listing 7. The function `inla3dt::interpretable2spde( ... )` converts the *interpretable model parameters* $(\sigma_e, \rho_s, \rho_t)$ into the *SPDE parameters* $\boldsymbol{\gamma} = (\gamma_t, \gamma_s, \gamma_e)$ and the logarithm thereof, $\boldsymbol{\theta} = (\theta_t, \theta_s, \theta_t) := (\ln(\gamma_t), \ln(\gamma_s), \ln(\gamma_e))$ as `$gamma` and `$theta` respectively; whereas the inverse is performed by `inla3dt::spde2interpretable( ... )` (cf. eqs. (2.17) to (2.19)). Both of these transformations are implemented for arbitrary DEMF models, and not only the DEMF$(1, 2, 1)$. To avoid bugs and confusion, we recommend passing arguments by name, and accessing the elements of `$gamma` and `$theta` by name as well (e.g. `$gamma[["s"]]` for γ_s).

4.5. Miscellaneous code snippets

We present here two miscellaneous code snippets, which the reader might find useful, but which do not warrant their own section, and which did not make it into any function.

[1]This was done when debugging the two versions—the `cgeneric` version is written in C, while the `rgeneric` version is written in R. Unsirprisingly, the `cgeneric` version is much faster

[2]Note, that while it is possible to retrieve the precision matrix $\tilde{\boldsymbol{Q}}_{st}$ from the `rgeneric` model using `INLA::inla.rgeneric.q( ... )`, the function `INLA::inla.cgeneric.q( ... )$Q` does not accept passing *SPDE parameters* as arguments, returning a precision matrix for (arbitrary) default parameters instead.

```r
spatial_A <- inlamesh3d::inla.mesh3d.make.A(spatial_mesh, loc)
A <- INLA::inla.spde.make.A(
        A.loc      = matrix(1, temporal_mesh$n) %x% spatial_A,
        group      = rep(temporal_mesh$loc, each = nrow(loc)),
        group.mesh = temporal_mesh
    ) %>%
    zapsmall() %>%
    drop0()
```

Listing 8: Creating the projector matrix $\boldsymbol{A}$. Let `spatial_mesh` and `temporal_mesh` be `INLA` mesh objects, and `loc` a vector of node indices at which observations are available. After creating the matrix, there might be some very small, non-zero entries— these are almost certainly numerical artefacts of the projection. Therefore they are set to zero (`zapsmall( ... )`), and then explicit zero entries are deleted from the matrix (`Matrix::drop0( ... )`), to increase its sparsity. We use the pipe operator (`%>%`) from `magrittr/tidyverse` purely for convenience.

First, in listing 8, we demonstrate how the projector matrix $\boldsymbol{A}$ can be constructed for spatio-temporal models. Next, in listing 9, we show how we can extract summary statistics for the linear predictor from the INLA output, and make them ready for export via `meshr::write_xdmf`. Finally, in listing 10, we do the same for the summary statistics of the fitted field $\boldsymbol{u}(\boldsymbol{x}, t)$,

```r
library(tidyverse)
eta <- est$summary.linear.predictor %>%
    rownames_to_column("var") %>%
    as_tibble() %>%
    mutate(
        kind = case_when(
            str_detect(var, "^APredictor\\.") ~ "A",
            str_detect(var, "^Predictor\\.")  ~ "meshpoint"
        ) %>% as.factor(),
        index = var %>% str_remove("^\\w+\\.") %>%
         ↪  as.integer()
    ) %>%
    select(
        kind,
        index,
        mean,
        sd,
        q_0.025 = `0.025quant`,
        q_0.5   = `0.5quant`,
        q_0.975 = `0.975quant`,
        d_kl    = kld
    ) %>%
    filter(kind == "meshpoint") %>%
    arrange(index)
```

.

Listing 9: Extracting summary statistics from the linear predictor. Here, `est` is the output from `INLA::inla( ... )`. The columns of `eta` can then be extracted and cast into matrices of adequate size, e.g. `matrix(eta$mean, n_nodes, n_timesteps)`, and passed as node attributes to `meshr::write_xdmf( ... )`

```r
library(tidyverse)
field ← est$summary.random$field %>%
  as_tibble() %>%
  select(
    index = ID,
    mean,
    sd,
    q_0.025 = `0.025quant`,
    q_0.5   = `0.5quant`,
    q_0.975 = `0.975quant`,
    d_kl    = kld
  ) %>%
  mutate(index = as.integer(index))
```

Listing 10: Analogous example to listing 9, for preparing the field summary statistics for export and plotting.

Chapter 5

Conclusion

In this thesis, we have shown how to fit 3D + time spatio-temporal models in R-INLA. We provide facilities for generating three-dimensional meshes in R, by providing interfaces to two well known meshers, TetGen [28] and Gmsh [11], and interactive visualisation in ParaView [25]; facilities which we find crucial for working with 3D + time models. Our testing procedure, as outlined in section 3.2, should prove valuable during the process of adding this functionality to official versions of R-INLA.

To conclude this work, we sketch what we envision might be a possible future for spatio-temporal modelling with R-INLA. First, we would welcome if functionalities from both inlamesh3d [16] and INLAspacetime [14] would be integrated into R-INLA, so that they are supported on par with other parts of the programme—to our knowledge, this is planned or already underway. Second, demonstration of more complex 3D + time models would be beneficial; as would the application to real-world datasets and problems—both of these would, in out opinion, show the viability of 3D + time modelling with R-INLA, and provide domain scientists with a stepping stone or foundation, upon which they can build their models. We consider MRI, meteorological and pollution data to be good first candidates for such a demonstration. It would also be interesting to compare the model performance of INLA to more physically informed models, e.g. how well can INLA model diffusion or heat transfer, compared to state-of-the-art numerical methods. Third, mesh generation could be improved, e.g. by adding support for ghost boundary cells in 3D to compensate for boundary effect artefacts. Since INLA is agnostic as to the underlying mesh topology, the use of meshes whose spatial structure changes through time could be explored, for example using the recently developed avro 4D-mesher [4, 5]. Finally, the impact of adding a third spatial dimension on performance is not clear, and might need further investigation.

Appendix A

Bibliography

[1] Haakon Bakka et al. *The Diffusion-Based Extension of the Matérn Field to Space-Time*. Version 1. June 8, 2020. DOI: 10.48550/arXiv.2006.04917v1. arXiv: 2006.04917 [stat]. URL: http://arxiv.org/abs/2006.04917v1 (visited on 2023-01-05).

[2] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. "The Quickhull Algorithm for Convex Hulls". In: *ACM Transactions on Mathematical Software (TOMS)* 22.4 (1996), pp. 469–483. DOI: 10.1145/235815.235821.

[3] Michela Cameletti et al. "Spatio-Temporal Modeling of Particulate Matter Concentration through the SPDE Approach". In: *AStA Advances in Statistical Analysis* 97.2 (Apr. 1, 2013), pp. 109–131. ISSN: 1863-818X. DOI: 10.1007/s10182-012-0196-3. URL: https://doi.org/10.1007/s10182-012-0196-3 (visited on 2022-11-15).

[4] Philip Claude Caplan. *Avro*. URL: https://gitlab.com/philipclaude/avro (visited on 2023-01-09).

[5] Philip Claude Caplan et al. "Four-Dimensional Anisotropic Mesh Adaptation". In: *Computer-Aided Design* 129 (2020), p. 102915. DOI: 10.1016/j.cad.2020.102915.

[6] Stuart Carson and Joanna Mills Flemming. "Seal Encounters at Sea: A Contemporary Spatial Approach Using R-INLA". In: *Ecological Modelling* 291 (Nov. 10, 2014), pp. 175–181. ISSN: 0304-3800. DOI: 10.1016/j.ecolmodel.2014.07.022. URL: https://www.sciencedirect.com/science/article/pii/S0304380014003597 (visited on 2023-01-08).

[7] Aurelie Cosandey-Godin et al. "Applying Bayesian Spatiotemporal Models to Fisheries Bycatch in the Canadian Arctic". In: *Canadian Journal of Fisheries and Aquatic Sciences* 72.2 (Feb. 2015), pp. 186–197. ISSN: 0706-652X. DOI: 10.1139/cjfas-2014-0159. URL: https://cdnsciencepub.com/doi/full/10.1139/cjfas-2014-0159 (visited on 2023-01-08).

[8] Geir-Arne Fuglstad et al. "Constructing Priors That Penalize the Complexity of Gaussian Random Fields". In: *Journal of the American Statistical Association* 114.525 (2019), pp. 445–452. DOI: 10.1080/01621459.2017.1415907.

[9] Lisa Gaedke-Merzhäuser et al. "Parallelized Integrated Nested Laplace Approximations for Fast Bayesian Inference". In: *Statistics and Computing* 33.1 (Dec. 24, 2022), p. 25. ISSN: 1573-1375. DOI: `10.1007/s11222-022-10192-1`. URL: `https://doi.org/10.1007/s11222-022-10192-1` (visited on 2023-01-09).

[10] Andrew Gelman et al. *Bayesian Data Analysis*. 3rd. New York, NY, USA: Chapman and Hall/CRC, 2013. 675 pp. ISBN: 978-0-429-11307-9. URL: `https://doi.org/10.1201/b16018`.

[11] Christophe Geuzaine and Jean-François Remacle. "Gmsh: A 3-D Finite Element Mesh Generator with Built-in Pre-and Post-Processing Facilities". In: *International journal for numerical methods in engineering* 79.11 (2009), pp. 1309–1331. DOI: `10.1002/nme.2579`.

[12] Virgilio Gómez-Rubio. *Bayesian Inference with INLA*. Boca Raton: Chapman & Hall/CRC, 2020. 330 pp. ISBN: 978-1-138-03987-2. URL: `http://becarioprecario.bitbucket.io/inla-gitbook` (visited on 2022-04-24).

[13] Rikke Ingebrigtsen et al. "Estimation of a Non-Stationary Model for Annual Precipitation in Southern Norway Using Replicates of the Spatial Field". In: *Spatial Statistics* 14 (Nov. 1, 2015), pp. 338–364. ISSN: 2211-6753. DOI: `10.1016/j.spasta.2015.07.003`. URL: `https://www.sciencedirect.com/science/article/pii/S2211675315000597` (visited on 2023-01-08).

[14] Elias T. Krainski. *INLAspacetime*. Dec. 8, 2022. URL: `https://github.com/eliaskrainski/INLAspacetime` (visited on 2022-12-11).

[15] Elias T. Krainski et al. *Advanced Spatial Modeling with Stochastic Partial Differential Equations Using R and INLA*. Boca Raton: Chapman & Hall/CRC, 2019. 298 pp. ISBN: 978-0-367-57064-4. URL: `https://becarioprecario.bitbucket.io/spde-gitbook/` (visited on 2022-04-24).

[16] Finn Lindgren. *Inlamesh3d*. Apr. 9, 2022. URL: `https://github.com/finnlindgren/inlamesh3d` (visited on 2022-12-21).

[17] Finn Lindgren, David Bolin, and Håvard Rue. "The SPDE Approach for Gaussian and Non-Gaussian Fields: 10 Years and Still Running". In: *Spatial Statistics* (Jan. 24, 2022), p. 100599. ISSN: 2211-6753. DOI: `10.1016/j.spasta.2022.100599`. URL: `https://www.sciencedirect.com/science/article/pii/S2211675322000057` (visited on 2022-04-24).

[18] Finn Lindgren, Håvard Rue, and Johan Lindström. "An Explicit Link between Gaussian Fields and Gaussian Markov Random Fields: The Stochastic Partial Differential Equation Approach". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.4 (2011), pp. 423–498. DOI: `10.1111/j.1467-9868.2011.00777.x`.

[19] Finn Lindgren et al. *A Diffusion-Based Spatio-Temporal Extension of Gaussian Matérn Fields*. Oct. 16, 2022. DOI: `10.48550/arXiv.2006.04917v2`. arXiv: `2006.04917 [stat]`. URL: `http://arxiv.org/abs/2006.04917v2` (visited on 2022-11-16).

[20] Bertil Matérn. *Spatial variation*. Report 49:5. Stockholm, 1960. 144 pp. URL: `https://pub.epsilon.slu.se/10033/` (visited on 2022-09-26).

[21] Richard McElreath. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. 2nd ed. CRC Texts in Statistical Science. Boca Raton: Taylor and Francis, CRC Press, 2020. ISBN: 978-0-367-13991-9.

[22] David L. Miller, Richard Glennie, and Andrew E. Seaton. "Understanding the Stochastic Partial Differential Equation Approach to Smoothing". In: *Journal of Agricultural, Biological and Environmental Statistics* 25.1 (Mar. 2020), pp. 1–16. ISSN: 1085-7117, 1537-2693. DOI: `10.1007/s13253-019-00377-z`. arXiv: `2001.07623 [stat]`. URL: `http://arxiv.org/abs/2001.07623` (visited on 2023-01-07).

[23] Paula Moraga et al. "Bayesian Spatial Modelling of Geostatistical Data Using INLA and SPDE Methods: A Case Study Predicting Malaria Risk in Mozambique". In: *Spatial and Spatio-temporal Epidemiology* 39 (Nov. 1, 2021), p. 100440. ISSN: 1877-5845. DOI: `10.1016/j.sste.2021.100440`. URL: `https://www.sciencedirect.com/science/article/pii/S1877584521000393` (visited on 2022-10-21).

[24] Duncam Murdoch. *RGL - 3D Visualization Device System for R Using OpenGL*. Dec. 2, 2022. URL: `https://github.com/dmurdoch/rgl` (visited on 2022-12-25).

[25] *ParaView*. KitWare Inc. URL: `https://gitlab.kitware.com/paraview/paraview` (visited on 2022-12-25).

[26] Håvard Rue, Sara Martino, and Nicolas Chopin. "Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 71.2 (Apr. 2009), pp. 319–392. ISSN: 13697412, 14679868. DOI: `10.1111/j.1467-9868.2008.00700.x`. URL: `https://onlinelibrary.wiley.com/doi/10.1111/j.1467-9868.2008.00700.x` (visited on 2022-09-26).

[27] Håvard Rue et al. "Bayesian Computing with INLA: A Review". In: *Annual Review of Statistics and Its Application* 4.1 (Mar. 7, 2017), pp. 395–421. ISSN: 2326-8298, 2326-831X. DOI: `10.1146/annurev-statistics-060116-054045`. URL: `https://www.annualreviews.org/doi/10.1146/annurev-statistics-060116-054045` (visited on 2023-01-09).

[28] Hang Si. "TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator". In: *ACM Transactions on Mathematical Software* 41.2 (Feb. 4, 2015), 11:1–11:36. ISSN: 0098-3500. DOI: `10.1145/2629697`. URL: `https://doi.org/10.1145/2629697` (visited on 2022-09-26).

[29] Per Sidén et al. *Spatial 3D Matérn Priors for Fast Whole-Brain fMRI Analysis*. Oct. 1, 2020. DOI: `10.48550/arXiv.1906.10591`. arXiv: `1906.10591 [stat]`. URL: `http://arxiv.org/abs/1906.10591` (visited on 2022-07-19).

[30] Michael L. Stein. *Interpolation of Spatial Data*. Springer Series in Statistics. New York, NY: Springer, 1999. ISBN: 978-1-4612-7166-6 978-1-4612-1494-6. DOI: `10.1007/978-1-4612-1494-6`. URL: `http://link.springer.com/10.1007/978-1-4612-1494-6` (visited on 2023-01-07).

[31] David C. Sterrat. *Geometry: Mesh Generation and Surface Tessellation*. In collab. with Jean-Romain Roussel et al. Version 0.4.6.1. July 4, 2022. URL: `http://davidcsterratt.github.io/geometry/` (visited on 2022-12-21).

[32] Michal Sudwoj. *Inla3dt*. Dec. 21, 2022. URL: `https://github.com/westernmagic/inla3dt` (visited on 2022-12-21).

[33] *The HDF5® Library & File Format*. The HDF Group. URL: `https://www.hdfgroup.org/solutions/hdf5/` (visited on 2023-01-11).

[34] Peter Whittle. "On Stationary Processes in the Plane". In: *Biometrika* (1954), pp. 434–449.

[35] *XDMF*. URL: `https://www.xdmf.org/index.php/Main_Page` (visited on 2022-12-25).

[36] *Xdmf HyperSlab with Timesteps - ParaView Support*. ParaView. Aug. 24, 2020. URL: `https://discourse.paraview.org/t/xdmf-hyperslab-with-timesteps/5174` (visited on 2023-01-11).

Appendix B

Glossary

API application programming interface. 13, 21, 22

CAD computer-aided design. 13

CLI command-line interface. 13

DEMF diffusion-based extension of the Gaussian Matérn field. 10, 16, 19, 26

FEM finite element method. 9, 11, 14, 36

GMRF Gaussian Markov random field. 7

GRF Gaussian random field. 7

HDF hierarchical data format. 14, 35

HDF5 hierarchical data format, version 5. 14, 24

INLA integrated nested Laplacian approximation. 2–4, 7, 9, 12–14, 20, 21, 23, 26, 27, 30

MCMC Markov chain Monte Carlo method. 4

MRF Markov random field. 7, 35

PDE partial differential equation. 8, 35

RF random field. 7, 35

spd symmetric positive-definite. 8

SPDE stochastic PDE. 2, 8–11, 16, 26

XDMF extensible data model and format. 14, 15, 21, 23, 24

XML extensible markup language. 14

Appendix C

Notation

- $\boldsymbol{x}$: vector
- $\boldsymbol{A}$: matrix
- $\mathbb{P}(x)$: probability of event x
- μ: mean
- σ: standard deviation
- σ^2: variance
- $\mathrm{cov}(x, y)$: covariance of x and y
- $\mathrm{cor}(x, y)$: correlation of x and y
- $\Gamma(z)$: Gamma function
- $\mathrm{K}_\alpha(z)$: modified Bessel function of the second kind of order α
- ψ_i: FEM basis function
- ϕ_i: FEM basis function
- $\langle f, g\rangle$: inner product
- $G \,\Box\, H$: definition 1
- $u \sim v$: u is adjacent to v
- K_n: complete graph of order n

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

Bayesian inference for 4D spatio-temporal models using Integrated nested Laplacian approximations

Verfasst von (in Druckschrift):
Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**
Sudwoj	Michal

Ich bestätige mit meiner Unterschrift:
- Ich habe keine im Merkblatt „Zitier-Knigge" beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**
Eglisau, 2023-01-13	[illegible]

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.