Journal of Statistical Software

September 2021, Volume 99, Issue 10. *doi: 10.18637/jss.v099.i10*

D-STEM v2: A Software for Modeling Functional Spatio-Temporal Data

Yaqiong Wang
Peking University

Francesco Finazzi
University of Bergamo

Alessandro Fassò
University of Bergamo

Abstract

Functional spatio-temporal data naturally arise in many environmental and climate applications where data are collected in a three-dimensional space over time. The MATLAB **D-STEM** v1 software package was first introduced for modeling multivariate space-time data and has been recently extended to **D-STEM** v2 to handle functional data indexed across space and over time. This paper introduces the new modeling capabilities of **D-STEM** v2 as well as the complexity reduction techniques required when dealing with large data sets. Model estimation, validation and dynamic kriging are demonstrated in two case studies, one related to ground-level air quality data in Beijing, China, and the other one related to atmospheric profile data collected globally through radio sounding.

Keywords: functional data analysis, 4D data, climate data, environmetrics, EM algorithm.

1. Introduction

With the increase of multidimensional data availability and modern computing power, statistical models for spatial and spatio-temporal data are developing at a rapid pace. Hence, there is a need for stable and reliable, yet updated and efficient, software packages. In this section, we briefly discuss multidimensional data in climate and environmental studies as well as statistical software for space-time data.

1.1. Multidimensional data

Large multidimensional data sets often arise when climate and environmental phenomena are observed at the global scale over extended periods. In climate studies, relevant physical variables are observed on a three-dimensional (3D) spherical shell (the atmosphere) while time is the fourth dimension. For instance, measurements are obtained by radiosondes flying from ground level up to the stratosphere (Fassò, Ignaccolo, Madonna, Demoz, and Franco-Villoria 2014), by interferometric sensors aboard satellites (Finazzi, Fassò, Madonna, Negri, Sun, and Rosoldi 2019a) or by laser-based methods, such as light detection and ranging (LIDAR; Negri, Fassò, Mona, Papagiannopoulos, and Madonna 2018). In this context, statistical modeling of multidimensional data requires describing and exploiting the spatio-temporal correlation of the underlying phenomenon or data-generating process. This is done using explanatory variables and multidimensional latent variables with covariance functions defined over a convenient spatio-temporal support. When considering 3D × T data (4D for brevity), covariance functions defined over the 4D support may be adopted. However, these covariance functions often have a complex form (Porcu, Alegria, and Furrer 2018). Moreover, when estimating the model parameters or making inferences, very large covariance matrices (though they may be sparse) are implied.

In large climate and environmental applications, 4D data are rarely collected at high frequency in all spatial and temporal dimensions. Often, only one dimension is sampled at high frequency while the remaining dimensions are sampled sparsely. Radiosonde data, for instance, are sparse over the Earth's sphere, but they are dense along the vertical dimension, providing atmospheric profiles. This suggests that handling all spatial dimensions equally (e.g., using a 3D covariance function) may not be the best option from a modeling or computational perspective, and a data reduction technique may be useful instead. In this paper, the functional data analysis (FDA) approach (Ramsay and Silverman 2007) is adopted to model the relationship between measurements along the profile, while the remaining dimensions are handled following the classic spatio-temporal data modeling approach using only 2D spatial covariance functions.

1.2. Statistical software

Various software programs are available for considering data on a plane or in a two-dimensional (2D) Euclidean space. The choice is more restricted when considering multidimensional or non-Euclidean spaces arising from atmospheric or remote sensing spatio-temporal data observed on the surface of a sphere and over time.

For example, Figure 1 depicts the spatial locations of measurements collected globally in a single day through radio sounding, as discussed in Section 5. Space is three-dimensional, and measurements are repeated over time at the same spatial locations over the Earth's surface but at different pressure values.

The **spBayes** package (Finley, Banerjee, and Gelfand 2015) handles large spatio-temporal data sets, but space is only 2D. The documentation of the **spacetime** (Pebesma 2012) and **gstat** (Pebesma and Heuvelink 2016) packages does not explicitly address the multidimensional case, but, according to Gasch, Hengl, Gräler, Meyer, Magney, and Brown (2015), both packages have some capabilities to handle the 3D × T setting. However, we want to avoid working with 3D spatial covariance functions or sample spatio-temporal variograms. Fixed rank kriging (Cressie and Johannesson 2008) implemented in the R (R Core Team 2021) package **FRK** (Zammit-Mangion and Cressie 2021; Zammit-Mangion and Sainsbury-Dale 2021) handles spatial and spatio-temporal data both on the Euclidean plane and on the surface of the sphere. **FRK** implements a set of tools for data gridding and basis function computation, resulting in efficient dimension reduction, allowing it to handle large satellite data sets (Cressie 2018). It is based on a spatio-temporal random effects (SRE) model estimated by the expectation-

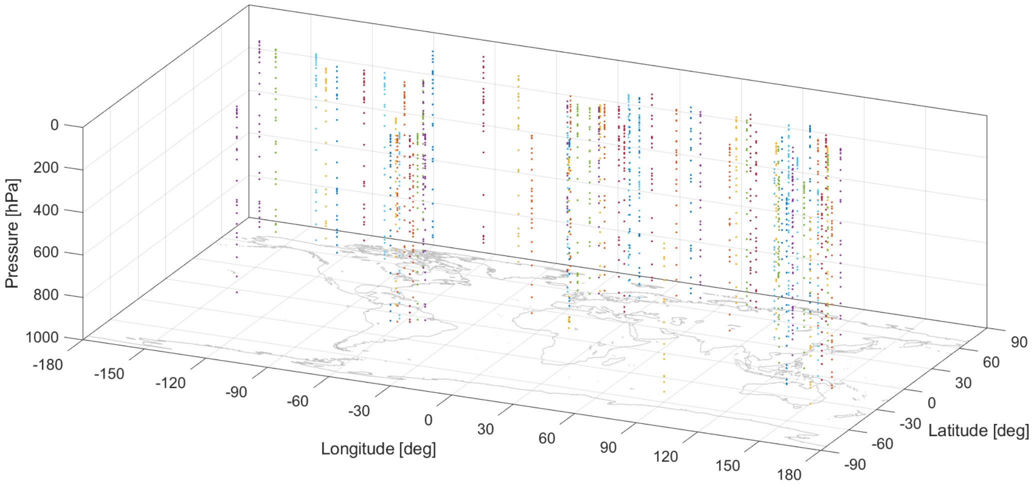

Figure 1: Radio sounding data example. Each dot represents the spatial location of a measurement taken by a radiosonde. Dots of the same color belong to the same radiosonde. (Pressure axis not in scale).

maximization (EM) algorithm. Recent extensions to **FRK** include the use of multi-resolution basis functions (Tzeng and Huang 2018).

A second package based on SRE and the EM algorithm is the **D-STEM** v1 package (Finazzi and Fassò 2014) for MATLAB (The MathWorks Inc. 2021). This package implements an efficient state-space approach for handling the temporal dimension and a heterotopic multivariate response approach that is useful when correlating heterogeneous networks (Fassò and Finazzi 2011; Calculli, Fassò, Finazzi, Pollice, and Turnone 2015).

D-STEM v1 has been successfully used in various medium-to-large applications, proving that the EM algorithm implementation, being mainly based on closed-form iterations, is quite stable. These applications include air quality assessment in the metropolitan areas of Milan, Teheran and Beijing (Fassò 2013; Taghavi-Shahri, Fassò, Mahaki, and Amin 2019; Wan, Xu, Huang, and Chen 2021); multivariate spatio-temporal modeling at the country and continental levels in Europe (Finazzi, Scott, and Fassò 2013; Fassò, Finazzi, and Ndongo 2016); time series clustering (Finazzi, Haggarty, Miller, Scott, and Fassò 2015); emulators of atmospheric dispersion modeling systems (Finazzi, Napier, Scott, Hills, and Cameletti 2019b); and near real-time prediction of earthquake parameters (Finazzi 2020).

A brief, non-exhaustive list of other models and/or software packages for advanced spatial data modeling is presented below, according to the principal technique, allowing the handling of large data sets. In general, these techniques aim at avoiding the Cholesky decomposition of large and dense covariance matrices.

Some approaches, including **FRK** and **D-STEM** v1, leverage sparse variance-covariance matrices. Others exploit the sparsity of the precision matrix, thanks to a spatial Markovian assumption. This class includes the R packages **LatticeKrig** (Nychka, Bandyopadhyay, Hammerling, Lindgren, and Sain 2015; Nychka, Hammerling, Sain, and Lenssen 2019), **INLA** (Blangiardo, Cameletti, Baio, and Rue 2013; Lindgren and Rue 2015; Bivand, Gómez-Rubio, and Rue 2015; Rue, Martino, Blangiardo, Simpson, Riebler, and Krainski 2014) and the multi-resolution approximation approach of Katzfuss (2017), which uses the predictive process and the state space representation (Jurek and Katzfuss 2021) to model spatio-temporal data. Low-rank models are another popular approach used by **spBayes**. Finally, the R package **laGP** (Gramacy 2016), based on a machine learning approach, implements an efficient nearest neighbor prediction-oriented method. Heaton *et al.* (2019) develop an interesting spatial prediction competition considering a large data set and involving the above-mentioned approaches.

We observe that, although some of the software packages mentioned above consider both space and time, to the best of our knowledge, none of them handles a spatio-temporal FDA approach for data sets of the kind discussed in Section 1.1.

In this paper, we present the MATLAB package **D-STEM** v2, extending **D-STEM** v1. The new version introduces modeling of functional data indexed over space and time. Moreover, new complexity reduction techniques have been added for both model estimation and dynamic mapping, which are especially useful for large data sets.

The rest of the paper is organized as follows. Section 2 introduces the methodology adopted in this paper and, in particular, the data modeling approach and the complexity-reduction techniques. Section 3 describes the **D-STEM** v2 software in terms of the MATLAB classes used to define the data structure, model fitting and diagnostics and kriging. This is followed by an illustration of the software use through two case studies. The first one, discussed in Section 4, considers high-frequency spatio-temporal ozone data in Beijing. The second one, in Section 5, considers modeling of global atmospheric temperature profiles and exploits the complexity-reduction capabilities of the new package. Finally, concluding remarks are provided in Section 6.

2. Methodology

This section discusses the methodology behind the modeling and the complexity-reduction techniques implemented in **D-STEM** v2 when dealing with functional space-time data sets. Moreover, model estimation, validation and dynamic kriging are briefly discussed.

2.1. Model equations

Let $\boldsymbol{s} = (s_{lat}, s_{lon})^\top$ be a generic spatial location on the Earth's sphere, $\mathbb{S}^2$, and $t \in \mathbb{N}$ a discrete time index. It is assumed that the function of interest, $f(\boldsymbol{s}, h, t)$, with domain $\mathcal{H} = [h_1, h_2] \subset \mathbb{R}$, can be observed at any $(\boldsymbol{s}, t)$ and $h \in \mathcal{H}$ through noisy measurements $y(\boldsymbol{s}, h, t)$ according to the following model:

$$y(\boldsymbol{s}, h, t) = f(\boldsymbol{s}, h, t) + \varepsilon(\boldsymbol{s}, h, t), \tag{1}$$

$$f(\boldsymbol{s}, h, t) = \boldsymbol{x}(\boldsymbol{s}, h, t)^\top \boldsymbol{\beta}(h) + \boldsymbol{\phi}(h)^\top \boldsymbol{z}(\boldsymbol{s}, t), \tag{2}$$

$$\boldsymbol{z}(\boldsymbol{s}, t) = \boldsymbol{G}\boldsymbol{z}(\boldsymbol{s}, t-1) + \boldsymbol{\eta}(\boldsymbol{s}, t). \tag{3}$$

This model is referred to as the functional hidden dynamic geostatistical model (f-HDGM). In Equation 1, ε is a zero-mean Gaussian measurement error independent in space and time with functional variance $\sigma_\varepsilon^2(h)$, implying that ε is heteroskedastic across the domain $\mathcal{H}$. The variance is modeled as

$$\log(\sigma_\varepsilon^2(h)) = \boldsymbol{\phi}(h)^\top \boldsymbol{c}_\varepsilon,$$

where $\boldsymbol{\phi}(h)$ is a $p \times 1$ vector of basis functions evaluated at h, while $\boldsymbol{c}_{\varepsilon}$ is a vector of coefficients to be estimated. In Equation 2, $\boldsymbol{x}(\boldsymbol{s},h,t)$ is a $b \times 1$ vector of covariates while $\boldsymbol{\beta}(h) = (\beta_1(h),\ldots,\beta_b(h))^\top$ is the vector of functional parameters modeled as

$$\beta_j(h) = \boldsymbol{\phi}(h)^\top \boldsymbol{c}_{\beta,j}, \qquad j = 1,\ldots,b,$$

and $\boldsymbol{c}_{\beta} = \left(\boldsymbol{c}_{\beta,1}^\top,\ldots,\boldsymbol{c}_{\beta,b}^\top\right)^\top$ is a $pb \times 1$ vector of coefficients that needs to be estimated. Additionally, $\boldsymbol{z}(\boldsymbol{s},t)$ is a $p \times 1$ latent space-time variable with Markovian dynamics given in Equation 3. The matrix $\boldsymbol{G}$ is a diagonal transition matrix with diagonal elements in the $p \times 1$ vector $\boldsymbol{g}$. The innovation vector $\boldsymbol{\eta}$ is obtained from a multivariate Gaussian process that is independent in time but correlated across space with matrix spatial covariance function given by

$$\boldsymbol{\Gamma}(\boldsymbol{s},\boldsymbol{s}';\boldsymbol{\theta}) = \operatorname{diag}\left(v_1\rho(\boldsymbol{s},\boldsymbol{s}';\boldsymbol{\theta}_1),\ldots,v_p\rho(\boldsymbol{s},\boldsymbol{s}';\boldsymbol{\theta}_p)\right),$$

where $\boldsymbol{v} = (v_1,\ldots,v_p)^\top$ is a vector of variances and $\rho(\boldsymbol{s},\boldsymbol{s}';\boldsymbol{\theta}_j)$ is a valid spatial correlation function for locations $\boldsymbol{s},\boldsymbol{s}' \in \mathbb{S}^2$, parameterized by $\boldsymbol{\theta}_j$, and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_p)^\top$. The unknown model parameter vector is given by $\boldsymbol{\psi} = \left(\boldsymbol{c}_{\varepsilon}^\top,\boldsymbol{c}_{\beta}^\top,\boldsymbol{g}^\top,\boldsymbol{v}^\top,\boldsymbol{\theta}^\top\right)^\top$.

Note that, in order to ease the notation, the same p-dimensional basis functions $\boldsymbol{\phi}(h)$ are used to model σ_{ε}^2, β_j and $\boldsymbol{\phi}(h)^\top\boldsymbol{z}(\boldsymbol{s},t)$ in Equations 1–3. In practice, **D-STEM** v2 allows one to specify a different number of basis functions for each model component. Also note that ε is not a pure measurement error since it also accounts for model misspecification. Finally, the covariates $\boldsymbol{x}(\boldsymbol{s},h,t)$ are assumed to be known without error for any $\boldsymbol{s}$, h and t, and thus they do not need a basis function representation.

2.2. Basis function choice

Choosing basis functions essentially means choosing the basis type and the number of basis functions. **D-STEM** v2 currently supports Fourier bases and B-spline bases. The former guarantee that the function is periodic in the domain $\mathcal{H}$, while the latter are not (in general) periodic but have higher flexibility in describing functions with a complex shape. Whichever basis function type is adopted, the number p of basis functions must be fixed before model estimation. Usually, a high p implies a better model R^2, but over-fitting may be an issue. Moreover, special care must be taken when choosing the number of basis functions for $\boldsymbol{\phi}(h)^\top\boldsymbol{z}(\boldsymbol{s},t)$. The classic FDA approach suggests fixing a high number of basis functions and adopting penalization to avoid over-fitting. In our context, this is not viable since the covariance matrices involved in model estimation have dimension $n^3p^3 \times n^3p^3$. Since n is usually large, a large p would make model estimation unfeasible, especially if the number of time points T is also high. When using the B-spline basis, a small p implies that the location of knots along the domain $\mathcal{H}$ also matters and may affect the model fitting performance. Ideally, p and knot locations are chosen using a model validation technique (see Section 2.7) by trying different combinations of p and knot locations. If, due to time constraints, this is not possible, equally spaced knots are a convenient option.

2.3. Model estimation

The estimation of $\boldsymbol{\psi}$ and the latent space-time variable $\boldsymbol{z}(\boldsymbol{s},t)$ is based on the maximum likelihood approach considering profile data observed at spatial locations $\mathcal{S} = \{\boldsymbol{s}_i, i = 1,\ldots,n\}$ and time points $t = 1,\ldots,T$.

At a specific location $\boldsymbol{s}_i$ and time t, $q_{i,t}$ measurements are taken at points $\boldsymbol{h}_{\boldsymbol{s}_i,t} = (h_{i,1,t}, \ldots, h_{i,q_{i,t},t})^\top$ and collected in the vector

$$\boldsymbol{y}_{\boldsymbol{s}_i,t} = (y(\boldsymbol{s}_i, h_{i,1,t}, t), \ldots, y(\boldsymbol{s}_i, h_{i,q_{i,t},t}, t))^\top,$$

here called the observed profile.

Although **D-STEM** v2 allows for varying $q_{i,t}$, for ease of notation, it is assumed here that all profiles include exactly q measurements, although $\boldsymbol{h}_{\boldsymbol{s}_i,t}$ may be different across profiles. Profiles observed at time t across spatial locations $\mathcal{S}$ are then stored in the $nq \times 1$ vector $\boldsymbol{y}_t = (\boldsymbol{y}_{s_1,t}^\top, \ldots, \boldsymbol{y}_{s_n,t}^\top)^\top$. Applying model (1)–(3) to the defined data above, we have the following matrix representation:

$$\begin{aligned} \boldsymbol{y}_t &= \tilde{\boldsymbol{X}}_t \boldsymbol{c}_\beta + \boldsymbol{\Phi}_{z,t} \boldsymbol{z}_t + \boldsymbol{\varepsilon}_t, \\ \boldsymbol{z}_t &= \tilde{\boldsymbol{G}} \boldsymbol{z}_{t-1} + \boldsymbol{\eta}_t, \end{aligned}$$

where $\tilde{\boldsymbol{X}}_t = \boldsymbol{X}_t \boldsymbol{\Phi}_{\beta,t}$ is a $nq \times bp$ matrix, with $\boldsymbol{X}_t$ the matrix of covariates and $\boldsymbol{\Phi}_{\beta,t}$ the basis matrix for $\boldsymbol{\beta}$. $\boldsymbol{\Phi}_{z,t}$ is the $nq \times np$ basis matrix for the latent $np \times 1$ vector $\boldsymbol{z}_t = (\boldsymbol{z}(\boldsymbol{s}_1, t)^\top, \ldots, \boldsymbol{z}(\boldsymbol{s}_n, t)^\top)^\top$. $\boldsymbol{\eta}_t = (\boldsymbol{\eta}(\boldsymbol{s}_1, t)^\top, \ldots, \boldsymbol{\eta}(\boldsymbol{s}_n, t)^\top)^\top$ is the $np \times 1$ innovation vector, while $\boldsymbol{\varepsilon}_t$ is the $nq \times 1$ vector of measurement errors. Additionally, $\tilde{\boldsymbol{G}} = \boldsymbol{I}_n \otimes \boldsymbol{G}$ is the $np \times np$ diagonal transition matrix.

The complete-data likelihood function $L(\boldsymbol{\psi}; \boldsymbol{Y}, \boldsymbol{Z})$ can be written as

$$L(\boldsymbol{\psi}; \boldsymbol{Y}, \boldsymbol{Z}) = L(\boldsymbol{\psi}_{\boldsymbol{z}_0}; \boldsymbol{z}_0) \prod_{t=1}^{T} L(\boldsymbol{\psi}_{\boldsymbol{y}}; \boldsymbol{y}_t | \boldsymbol{z}_t) L(\boldsymbol{\psi}_{\boldsymbol{z}}; \boldsymbol{z}_t | \boldsymbol{z}_{t-1}),$$

where $\boldsymbol{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T)$, $\boldsymbol{Z} = (\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_T)$, $\boldsymbol{\psi}_{\boldsymbol{z}} = \left(\boldsymbol{g}^\top, \boldsymbol{v}^\top, \boldsymbol{\theta}^\top\right)^\top$, $\boldsymbol{\psi}_{\boldsymbol{y}} = \left(\boldsymbol{c}_\varepsilon^\top, \boldsymbol{c}_\beta^\top\right)^\top$, and $\boldsymbol{z}_0$ is the Gaussian initial vector with parameter $\boldsymbol{\psi}_{\boldsymbol{z}_0}$. Maximum likelihood estimation is based on an extension of the EM algorithm detailed in Calculli *et al.* (2015). The model parameter set $\boldsymbol{\psi}$ is initialized with starting values $\boldsymbol{\psi}^{\langle 0 \rangle}$ and then updated at each iteration ι of the EM algorithm.

The algorithm terminates if any of the following conditions is satisfied:

$$\max_l \left| \psi_l^{\langle \iota \rangle} - \psi_l^{\langle \iota-1 \rangle} \right| / \left| \psi_l^{\langle \iota \rangle} \right| < \epsilon_1$$

$$\left| L(\boldsymbol{\psi}^{\langle \iota \rangle}; \boldsymbol{Y}) - L(\boldsymbol{\psi}^{\langle \iota-1 \rangle}; \boldsymbol{Y}) \right| / \left| L(\boldsymbol{\psi}^{\langle \iota \rangle}; \boldsymbol{Y}) \right| < \epsilon_2,$$

$$\iota > \iota^*,$$

where $\psi_l^{\langle \iota \rangle}$ is the generic element of $\boldsymbol{\psi}^{\langle \iota \rangle}$ at the ι-th iteration, $L(\boldsymbol{\psi}^{\langle \iota \rangle}; \boldsymbol{Y})$ is the observed-data likelihood function evaluated at $\boldsymbol{\psi}^{\langle \iota \rangle}$, $0 < \epsilon_1 \ll 1$ and $0 < \epsilon_2 \ll 1$ are small positive numbers (e.g., 10^{-4}), while ι^* is a user-defined positive integer number (e.g., 100) to limit the iterations in the case of convergence failure of the EM algorithm.

Note that $\mathcal{S}$ is not time-varying, which means that spatial locations are fixed. This could be a limit in applications where spatial locations change for each t. On the other hand, missing profiles are allowed; that is, $\boldsymbol{y}_{\boldsymbol{s}_i,t}$ may be a vector of q missing values at some t. In the extreme case, a given spatial location $\boldsymbol{s}_i$ has only one profile over the entire period (if all the profiles are missing, the spatial location can be dropped from the data set). Shumway and Stoffer (2017, p. 348) explain how the likelihood function of a state-space model changes in the case of a missing observation vector and how the EM estimation formulas are derived. Missing data handling in **D-STEM** v2 is based on the same approach.

2.4. Partitioning

At each iteration of the EM algorithm, the computational complexity of the E-step is equal to $\mathcal{O}\left(Tn^3p^3\right)$, which may be unfeasible if n is large. When necessary, **D-STEM** v2 allows one to use a partitioning approach (Stein 2013) for model estimation. The spatial locations $\mathcal{S}$ are divided into k partitions, and $\boldsymbol{z}_t$ is partitioned conformably, namely, $\boldsymbol{z}_t = \left(\boldsymbol{z}_t^{(1)\top}, \ldots, \boldsymbol{z}_t^{(k)\top}\right)^\top$. Hence, the likelihood function becomes

$$\prod_{t=1}^{T} L\left(\boldsymbol{\psi}_{\boldsymbol{y}}; \boldsymbol{y}_t \mid \boldsymbol{z}_t\right) \cdot \prod_{j=1}^{k} L\left(\boldsymbol{\psi}_{\boldsymbol{z}_0}; \boldsymbol{z}_0^{(j)}\right) \cdot \prod_{j=1}^{k}\prod_{t=1}^{T} L\left(\boldsymbol{\psi}_{\boldsymbol{z}}; \boldsymbol{z}_t^{(j)} \mid \boldsymbol{z}_{t-1}^{(j)}\right).$$

From the point of view of the EM algorithm, this implies that the E-step is independently applied to each partition, possibly in parallel. When all partitions are equal in size, the computational complexity reduces to $\mathcal{O}\left(Tkr^3p^3\right)$, where r is the partition size.

Geographical partitioning, constructed by aggregating proximal locations, is a natural choice for environmental applications. Given the number of partitions k, the k-means algorithm applied to spatial coordinates provides a geographical partitioning of $\mathcal{S}$. However, the number of points in each partition is not controlled, and a heterogeneous partitioning may arise. If some subsets are very large and others are small, the reduction in computational complexity given above is far from being achieved. This can easily happen, for example, when $\mathcal{S}$ is a global network constrained by continent shapes.

For this reason, **D-STEM** v2 provides a heuristically modified k-means algorithm that encourages partitions with similar numbers of elements and which is based on the geodesic distance. The algorithm optimizes the following objective function:

$$\sum_{j=1}^{k}\sum_{\boldsymbol{s}\in\mathcal{S}_j} d\left(\boldsymbol{s}, \boldsymbol{c}_j\right) + \lambda \sum_{j=1}^{k}\left(r_j - \frac{n}{k}\right)^2, \tag{4}$$

where $\lambda \geq 0$, $\mathcal{S}_j \subset \mathcal{S}$ is the set of coordinates in the j-th partition, d is the geodesic distance on the sphere $\mathbb{S}^2$ and $\boldsymbol{c}_j$ and r_j are the centroid and the number of elements in the j-th partition, respectively. The second term in (4) accounts for the variability of the partition sizes and acts as a penalization for heterogeneous partitionings. Clearly, when $\lambda = 0$, the above-mentioned objective function is similar to the classic k-means algorithm based on the Euclidean distance. For high values of λ, solutions with similarly sized partitions are favored.

Unfortunately, an optimality theory for this algorithm has not yet been developed, and the choice of λ is left to the user. Nonetheless, it may be a useful tool to define a partitioning that is appropriate for the application at hand with regard to computing time and geographical properties.

2.5. Variance-covariance matrix estimation

The EM algorithm provides a point estimate of the parameter vector $\boldsymbol{\psi}$ but no uncertainty information. Building on Shumway and Stoffer (2017, p. 408), **D-STEM** v2 estimates the variance-covariance matrix $\Sigma_{\boldsymbol{\psi},T} = \mathbb{V}\left(\boldsymbol{\psi} \mid \boldsymbol{Y}\right)$, by means of the observed Fisher information matrix, $\mathbf{I}_T$, namely

$$\hat{\Sigma}_{\boldsymbol{\psi},T} = \left(\mathbf{I}_T\right)^{-1}.$$

To understand its computational cost, note that the information matrix given above may be written as a sum: $\mathbf{I}_T = \sum_{t=1}^{T} \mathbf{i}_t$.

For large data sets, each matrix $\mathbf{i}_t$ may be expensive to compute, and the total computational cost is linear in T, provided missing data are evenly distributed in time. This results in a time-consuming task with a computational burden even higher than that for model estimation. For this reason, **D-STEM** v2 makes it possible to approximate $\hat{\Sigma}_{\boldsymbol{\psi},T}$ using a truncated information matrix, namely:

$$\tilde{\Sigma}_{\boldsymbol{\psi},t^*} = \left(\frac{T}{t^*}\mathbf{I}_{t^*}\right)^{-1}, \tag{5}$$

which reduces the computational burden by a factor of $1 - t^*/T$.

Since $\tilde{\Sigma}_{\boldsymbol{\psi},t^*} \to \hat{\Sigma}_{\boldsymbol{\psi},T}$ for $t^* \to T$, the truncation time t^* is chosen to control the approximation error in $\hat{\Sigma}_{\boldsymbol{\psi},t}$. In particular, t^* is the first integer t such that

$$\frac{\left\|\tilde{\Sigma}_{\boldsymbol{\psi},t} - \tilde{\Sigma}_{\boldsymbol{\psi},t-1}\right\|_F}{\left\|\tilde{\Sigma}_{\boldsymbol{\psi},t}\right\|_F} \leq \delta, \tag{6}$$

where $\|\cdot\|_F$ is the Frobenius norm, and δ may be defined by the user.

Generally speaking, the behavior of $\hat{\Sigma}_{\boldsymbol{\psi},T}$ for large T and, hence, the behavior of $\tilde{\Sigma}_{\boldsymbol{\psi},t^*}$ relies on stationarity and ergodicity of the underlying stochastic process; see, for example, Shumway and Stoffer (2017, Property P6.4) and references therein.

To have operative guidance for the user, let us assume first that no missing values are present, the information matrix is well-conditioned and the covariates have no isolated outliers or extreme trends. In this case, away from the borders $t \cong 1$ and $t \cong T$, the observed conditional information $\mathbf{i}_t$ has a relatively smooth stochastic behavior, and the approximation in (5) is expected to be satisfactory at the level defined by δ. Conversely, if some data are missing at time t, the information $\mathbf{i}_t$ is reduced accordingly. If the missing pattern is random over time, this is not an issue. But, in the unfavorable case with a high percentage of missing data mostly concentrated at the end of the time series, $t \cong T$, the above approximation may over-estimate the information and under-estimate the variances of the parameter estimates.

2.6. Dynamic kriging

In this paper, dynamic kriging refers to evaluating the following quantities:

$$\hat{f}\left(\boldsymbol{s}, h, t\right) = \mathbb{E}_{\hat{\boldsymbol{\psi}}}\left(f\left(\boldsymbol{s}, h, t\right) \mid \boldsymbol{Y}\right), \tag{7}$$

$$\mathsf{VAR}\left(\hat{f}\left(\boldsymbol{s}, h, t\right)\right) = \mathbb{V}_{\hat{\boldsymbol{\psi}}}\left(f\left(\boldsymbol{s}, h, t\right) \mid \boldsymbol{Y}\right), \tag{8}$$

for any $\boldsymbol{s} \in \mathbb{S}^2$, $h \in \mathcal{H}$ and $t = 1, \ldots, T$. A common approach is to map the kriging estimates on a regular pixelation $\mathcal{S}^* = \{\boldsymbol{s}_1^*, \ldots, \boldsymbol{s}_m^*\}$. This may be a time-consuming task when m and/or n and/or T are large. To tackle this problem, **D-STEM** v2 allows one to exploit a nearest-neighbor approach, where the conditioning term in Equations 7 and 8 is not $\boldsymbol{Y}$, but the data at the spatial locations $\mathcal{S}_{\sim j}$, where $\mathcal{S}_{\sim j} \subset \mathcal{S}$ is the set of the $\tilde{n} \ll n$ nearest spatial locations to $\boldsymbol{s}_j^*$. The use of the nearest-neighbor approach is justified by the so-called screening effect. Even when the spatial correlation function exhibits long-range dependence, it can subsequently be assumed that y at spatial location $\boldsymbol{s}$ is nearly independent of spatially distant observations when conditioned on nearby observations (see Stein 2002; Furrer, Genton, and Nychka 2006, for more details).

For computational efficiency, **D-STEM** v2 performs kriging for blocks of pixels. To do this, $\mathcal{S}^*$ is partitioned in u blocks $\mathcal{S}^* = \{\mathcal{S}_1^*, \ldots, \mathcal{S}_u^*\}$, and kriging is done on each block $\mathcal{S}_l^*$, $l = 1, \ldots, u$, with $u \ll m$ controlled by the user. For each target block $\mathcal{S}_l^*$, the conditioning term in Equations 7 and 8 is given by the data observed at $\tilde{\mathcal{S}}_l = \bigcup_{j \in \mathcal{J}_l} \mathcal{S}_{\sim j}, \mathcal{J}_l = \left\{ j : s_j^* \in \mathcal{S}_l^* \right\}$. Note that, if $\mathcal{S}_l^*$ is a square or rectangular block of nearby pixels and $\mathcal{S}$ is sparse (namely $n \ll m$), then $\tilde{\mathcal{S}}_l$ is not much larger than $\mathcal{S}_{\sim j}$ since most of the spatial locations in $\mathcal{S}_l^*$ tend to have the same neighbors $\mathcal{S}_{\sim j}$.

2.7. Validation

D-STEM v2 allows one to implement an out-of-sample validation by partitioning the original spatial locations $\mathcal{S}$ into subsets $\mathcal{S}_{est}$ and $\mathcal{S}_{val}$. Data at $\mathcal{S}_{est}$ are used for model estimation while data at $\mathcal{S}_{val}$ are used for validation. Once the model is estimated, the kriging formula in Equation 7 is used to predict at $\mathcal{S}_{val}$ for all times t and heights $\boldsymbol{h}$. The following validation mean squared errors are then computed

$$MSE_t = \frac{1}{P_1} \sum_{\boldsymbol{s} \in \mathcal{S}_{val}} \sum_{h \in \boldsymbol{h}_{\boldsymbol{s},t}} \left(y\left(\boldsymbol{s}, h, t\right) - \hat{y}\left(\boldsymbol{s}, h, t\right) \right)^2,$$

$$MSE_{\boldsymbol{s}} = \frac{1}{P_2} \sum_{t=1}^{T} \sum_{h \in \boldsymbol{h}_{\boldsymbol{s},t}} \left(y\left(\boldsymbol{s}, h, t\right) - \hat{y}\left(\boldsymbol{s}, h, t\right) \right)^2,$$

$$MSE_h = \frac{1}{P_3} \sum_{t=1}^{T} \sum_{\boldsymbol{s} \in \mathcal{S}_{val}} \left(y\left(\boldsymbol{s}, h, t\right) - \hat{y}\left(\boldsymbol{s}, h, t\right) \right)^2,$$

where $\hat{y}\left(\boldsymbol{s}, h, t\right)$ is obtained from Equation 7, while P_1, P_2 and P_3 are the number of terms in each sum.

When $\boldsymbol{h}_{\boldsymbol{s},t}$ varies across the profiles, **D-STEM** v2 provides a binned MSE by splitting the continuous domain $\mathcal{H}$ into R equally spaced intervals. Let H_r^* be the set of observation points in the r-th interval, let n_r be the corresponding observation number and let $\bar{h}_r = \frac{1}{n_r} \sum_{h \in H_r^*} h$ be the mean of points in r-th interval. Then, the $MSE_{\bar{h}_r}$ is computed by

$$MSE_{\bar{h}_r} = \frac{1}{P_4} \sum_{h \in H_r^*} \sum_{t=1}^{T} \sum_{\boldsymbol{s} \in \mathcal{S}_{val}} \left(y\left(\boldsymbol{s}, h, t\right) - \hat{y}\left(\boldsymbol{s}, h, t\right) \right)^2,$$

where P_4 is the total number of observations in the r-th interval, $r = 1, ..., R$.

D-STEM v2 also provides the validation R^2 with respect to time

$$R_t^2 = 1 - \frac{MSE_t}{\mathsf{VAR}\left(\left\{ y\left(\boldsymbol{s}, h, t\right), \boldsymbol{s} \in \mathcal{S}_{val}, h \in \boldsymbol{h}_{\boldsymbol{s},t} \right\} \right)}.$$

and the analogous validation R^2 with respect to location $\boldsymbol{s}$ and h_r.

3. Software

This section starts by briefly describing the modeling capabilities of **D-STEM** v2 inherited by the previous version for dealing with spatio-temporal data sets. Then, it focuses on the **D-STEM** v2 classes and methods, which implement estimation, validation and dynamic mapping of the model presented in Section 2. Although some of the classes are already available in **D-STEM** v1, they are listed here for completeness.

3.1. Software description

D-STEM v1 implemented a substantial number of models. The dynamic co-regionalization model (DCM, Finazzi and Fassò 2014) and the hidden dynamic geostatistical model (HDGM, Calculli *et al.* 2015) are suitable for modeling and mapping multivariate space-time data collected from unbalanced monitoring networks. Model-based clustering (MBC, Finazzi *et al.* 2015) has been introduced for clustering time series, and it is suitable for large data sets with spatially registered time series. Moreover, the emulator model (Finazzi *et al.* 2019b) is based on a Gaussian emulator, and it is exploited for modeling the multivariate output of a complex physical model.

In addition, **D-STEM** v2 (Finazzi, Wang, and Fassó 2021) provides the functional version of HDGM, denoted by f-HDGM, which handles modeling and mapping of functional space-time data, following the methodology of Section 2. For implementing f-HDGM, **D-STEM** v2 relies on the MATLAB version of the **fda** package (Ramsay 2020), which is automatically downloaded and installed by **D-STEM** v2.

3.2. Data format

Two data formats are available to define observations for the f-HDGM. One is the internal format used by the **D-STEM** v2 classes, and the other one is the user format based on the more user-friendly '`table`' data type implemented in recent versions of MATLAB. The latter permits storing measurement profiles, covariate profiles, coordinates, timestamps and units of measure in a single object, though it prevents **D-STEM** v2 to run on Octave (Eaton, Bateman, Hauberg, and Wehbring 2020). The internal format is not discussed here.

Considering a table in the user format, each row includes the profiles collected at a given spatial location and time point. The column labels are defined as follows: columns `Y` and `Y_name` are used for the dependent variable y and its name as a string field, respectively; the column with prefix `X_h_` is used for the values of the domain h; eventually, columns with prefix `X_beta_` are used for covariates $\boldsymbol{x}$. These tables have only one column for y and only one column for h. Instead, we can have any number $b \geqslant 0$ of covariate columns. Additionally, the table has columns `X_coordinate` and `Y_coordinate` for spatial location $\boldsymbol{s}$ and column `Time` for the timestamp. Units of measure are stored in the `Properties.VariableUnits` property of the table columns and used in outputs and plots. Units for `X_coordinate` and `Y_coordinate` can be `deg` for degrees, `m` for meters and `km` for kilometers. Geodetic distance is used when the unit is `deg`; otherwise, the Euclidean distance is used.

At the table row corresponding to location $\boldsymbol{s}_i$ and time t, the elements related to y and $\boldsymbol{x}$ are vectors with $q_{i,t}$ elements. Vectors related to y may include missing data (`NaN`). If y is entirely missing for a given $(\boldsymbol{s}, t)$, the row must be removed from the table. Since spatial locations $\mathcal{S}$ are fixed in time, and as their number n is determined by the number of unique coordinates in the table, profiles observed at different time points but the same spatial location $\boldsymbol{s}$ must have the same coordinates.

3.3. Software structure

In **D-STEM**, a hierarchical structure of object classes and methods is used to handle data definition, model definition and estimation, validation, dynamic kriging and the related plotting capabilities. The structure is schematically given below. Further details on the use of each class are given within the two case studies in this paper, while class constructors, methods and property details can be obtained in MATLAB using the command

```
doc <class_name>
```

Data handling

The ‘stem_data’ class allows the user to define the data used in f-HDGM models, mainly through the following objects and methods.

- Objects of ‘stem_data’:
  - stem_modeltype: model type (DCM, HDGM, MBC, Emulator or f-HDGM); note that model type is needed here because the data structure varies among the different models;
  - stem_fda: basis functions specification;
  - stem_validation (optional): definition of the learning and testing data sets for model validation.
- Methods and properties of ‘stem_data’:
  - kmeans_partitioning: data partitioning for parallel EM computations of Section 2.4; this method is applied to a ‘stem_data’ object, and its output is used by the EM_estimation method in the ‘stem_model’ class below;
  - shape (optional): structure with geographical borders used for mapping.
- Internal objects of ‘stem_data’:
  - stem_varset: observed data and covariates;
  - stem_gridlist: list of ‘stem_grid’ objects
    - stem_grid: spatial locations coordinates;
  - stem_datestamp: temporal information.

Interestingly, stem_misc.data_formatter is a helper method, which is useful for building ‘stem_varset’ objects starting from data tables. Class ‘stem_misc’ provides additional methods for various intermediate tasks not discussed here for brevity.

Model building

The ‘stem_model’ class is used to define, estimate, validate and output a f-HDGM, mainly through the following objects and methods.

- Objects of ‘stem_model’
  - stem_data: defined above;
  - stem_par: model parameters;
  - stem_EM_result: container of the estimation output, after EM_estimate;
  - stem_validation_result (optional): container of validation output, available only if stem_data contains the stem_validation object;
  - stem_EM_options (optional): model estimation options; it is an input of the EM_estimate method below.
- Methods of ‘stem_model’
  - EM_estimate: computation of parameter estimates;
  - set_varcov: computation of the estimated variance-covariance matrix;
  - plot_profile: plot of functional data;
  - print: print estimated model summary;
  - beta_Chi2_test: testing significance of covariates;
  - plot_par: plot functional parameter;
  - plot_validation: plot MSE validation.

Kriging

The kriging handling is implemented with two classes. The first is the ‘stem_krig’ class, which implements the kriging spatial interpolation.

- Objects of ‘stem_krig’
  - stem_krig_data: mesh data for kriging;
  - stem_krig_options: kriging options;
- Methods of ‘stem_krig’
  - kriging: computation of kriging, the output is a ‘stem_krig_result’ object.

The second is the ‘stem_krig_result’ class, which stores the kriging output and implements the methods for plotting the kriging output.

- Methods of ‘stem_krig_result’
  - surface_plot: mapping of kriging estimate and their standard deviation for fixed h;
  - profile_plot: method for plotting the kriging function and the variance-covariance matrix for a fixed space and time.

Although at first reading the user could prefer a single object for both input and output of the kriging, these objects may be quite large, making the current approach more flexible.

4. Case study on ozone data

This section illustrates how to make inferences on an f-HDGM for ground-level high-frequency air quality data collected by a monitoring network. In particular, hourly ozone (O_3, in $\mu g/m^3$) measured in Beijing, China, is considered.

4.1. Air quality data

Ground-level O_3 is an increasing public concern due to its essential role in air pollution and climate change. In China, O_3 has become one of the most severe air pollutants in recent years (Wang, Xue, Brimblecombe, Lam, Li, and Zhang 2017).

In this case study, the aim is to model hourly O_3 concentrations from 2015 to 2017 with respect to temperature and ultraviolet radiation (UVB) across Beijing. Concentration and temperature data are available at twelve monitoring stations (Figure 2). Hourly UVB data are obtained from the ERA-Interim product of the European Centre for Medium-Range Weather Forecasts (ECMWF) at a grid size of 0.25° × 0.25° over the city.

To describe the diurnal cycle of O_3, which peaks in the afternoon and reaches a minimum at night-time, the 24 hours of the day are used as domain $\mathcal{H}$ of the basis functions, while the time index t is on the daily scale. Moreover, due to the circularity of time, Fourier basis functions are adopted, which implies that $\beta_j\left(h\right)$, $\sigma_\varepsilon^2\left(h\right)$ are periodic functions.

The measurement equation for O_3 is

$$y(\boldsymbol{s},h,t) = \beta_0(h) + x_{temp}(\boldsymbol{s},h,t)\beta_{temp}(h) + x_{uvb}\left(t\right)\beta_{uvb}(h) + \boldsymbol{\phi}(h)^\top \boldsymbol{z}(\boldsymbol{s},t) + \varepsilon(\boldsymbol{s},h,t), \quad (9)$$

where $\boldsymbol{s}$ is the generic spatial location, $h \in [0,24)$ is the time within the day expressed in hours and $t = 1,\ldots,1096$ is the day index over the period 2015–2017. Based on a preliminary

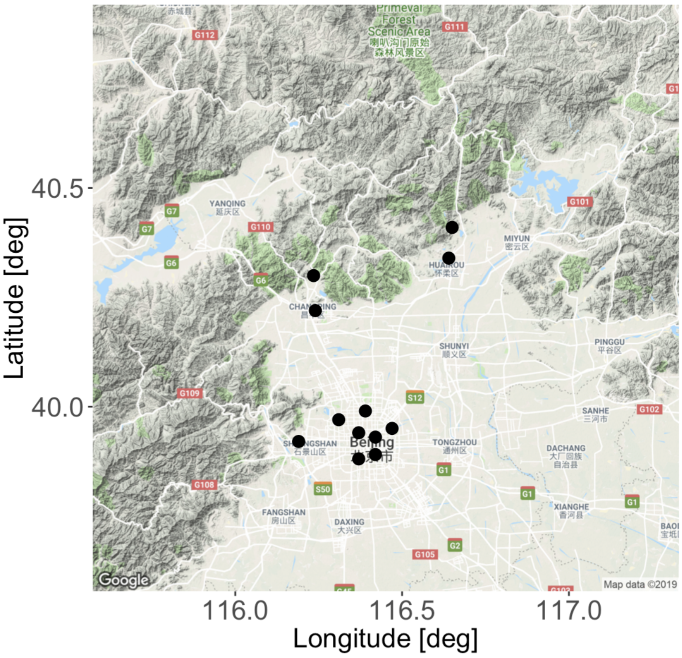

Figure 2: Spatial locations of the twelve stations in Beijing (Kahle and Wickham 2013).

analysis, the number of basis functions for $\beta_j(h)$, $\sigma_\varepsilon^2(h)$ and $\boldsymbol{\phi}(h)^\top \boldsymbol{z}(\boldsymbol{s},t)$ is chosen to be 5, 5 and 7, respectively.

4.2. Software implementation

This paragraph details the implementation of the **D-STEM** v2 in three aspects: model estimation, validation and kriging. Relevant scripts are `demo_section4_model_estimate.m`, `demo_section4_validation.m` and `demo_section4_kriging.m`, respectively, which are available in the supplementary material. All the scripts can be executed by choosing the option number from 1 to 3 in the `code.m` script.

Model estimation

This paragraph describes the `demo_section4_model_estimate.m` script devoted to the estimation of the model parameters and of their variance-covariance matrix.

The data set needed to perform this case study is stored as a MATLAB table in the user format of Section 3.2 and is named `Beijing_O3`. It can be loaded from the corresponding file as follows:

```
load Data/Beijing_O3
```

In the `Beijing_O3` table, each row refers to a fixed space-time point and gives a 24-element hourly ozone profile with the corresponding conformable covariates, which are: a constant, temperature and UVB.

The following lines of code specify the model type and the basis functions, which are stored in an object of class ‘`stem_fda`’:

```
o_modeltype = stem_modeltype('f-HDGM');
input_fda.spline_type = 'Fourier';
input_fda.spline_range = [0 24];
input_fda.spline_nbasis_z = 7;
input_fda.spline_nbasis_beta = 5;
input_fda.spline_nbasis_sigma = 5;
o_fda = stem_fda(input_fda);
```

When using a Fourier basis, `spline_nbasis_z` must be set to a positive odd number. Meanwhile, `spline_nbasis_beta` and/or `spline_nbasis_sigma` must be left empty, if $\boldsymbol{\beta}(h) \equiv \boldsymbol{\beta}$ and/or $\sigma_\varepsilon^2(h) \equiv \sigma_\varepsilon^2$ are constant functions.

The next step is to define an object of class ‘`stem_data`’, which specifies the model type and contains the basis function object and the data from the `Beijing_O3` table, transformed in the internal data format. This is done using the intermediate `input_data` structure:

```
input_data.stem_modeltype = o_modeltype;
input_data.data_table = Beijing_O3;
input_data.stem_fda = o_fda;
o_data = stem_data(input_data);
```

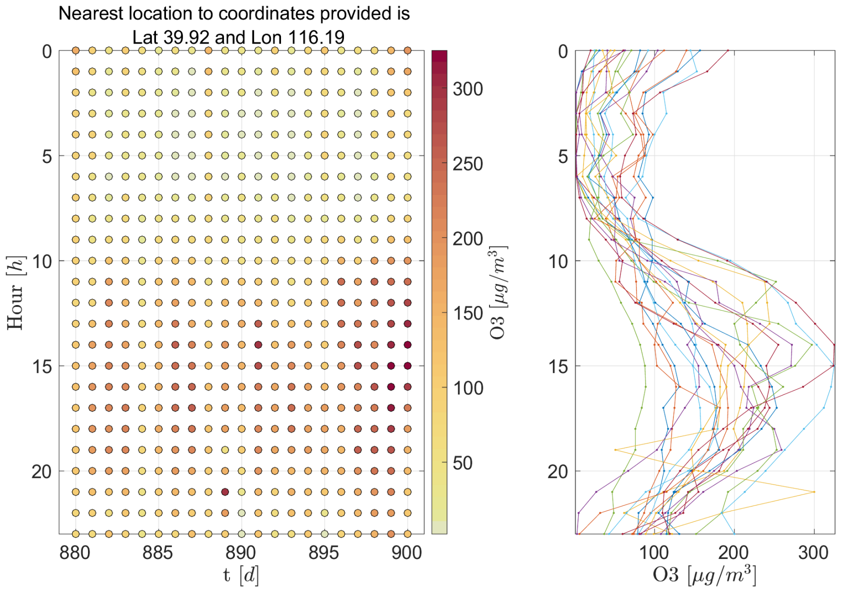

Figure 3: O_3 concentrations at location 39.92 latitude and 116.19 longitude for 21 days beginning on 29 May 2017. Left: each dot is a concentration measurement. The color of the dot depicts the concentration. Right: each graph is a daily concentration profile.

Then, an object of class ‘`stem_model`’ is created by using both information on data, stored in the `o_data` object, and on parameterization, contained in the ‘`stem_par`’ object named `o_par`:

```
o_par = stem_par(o_data, 'exponential');
o_model = stem_model(o_data, o_par);
```

To facilitate visualization, the method `plot_profile` of class ‘`stem_model`’ shows the O_3 profile data at location (`lat0`, `lon0`), in the days between `t_start` and `t_end` (Figure 3):

```
lat0 = 40; lon0 = 116;
t_start = 880; t_end = 900;
o_model.plot_profile(lat0, lon0, t_start, t_end);
```

Before running the EM algorithm, the model parameters need to be initialized. This is done using the method `get_beta0` of class ‘`stem_model`’, which provides the starting values for $\boldsymbol{\beta}$, and the method `get_coe_log_sigma_eps0` for the case of a functional $\sigma_\varepsilon^2(h)$. Next, the method `set_initial_values` of the `o_model` object is called to complete the initialization of model parameters:

```
n_basis = o_fda.get_basis_number;
o_par.beta = o_model.get_beta0();
o_par.sigma_eps = o_model.get_coe_log_sigma_eps0();
o_par.theta_z = ones(1, n_basis.z) * 0.18;
o_par.G = eye(n_basis.z) * 0.5;
o_par.v_z = eye(n_basis.z) * 10;
o_model.set_initial_values(o_par);
```

	χ^2 statistic	p value
Constant	136.3425	$< 10^{-16}$
Temperature	14265.6701	$< 10^{-16}$
UVB	2094.3523	$< 10^{-16}$

Table 1: χ^2 tests for significance of covariates.

Note that the `theta_z` parameter must be provided in the same unit of measure as the spatial coordinates.

Before model estimation, EM exiting conditions ϵ_1 (`exit_toll_par`), ϵ_2 (`exit_toll_loglike`) and ι^* (`max_iterations`) introduced in Section 2.3 can be optionally defined as follows:

```
o_EM_options = stem_EM_options();
o_EM_options.exit_toll_par = 0.0001;
o_EM_options.exit_toll_loglike = 0.0001;
o_EM_options.max_iterations = 200;
```

Model estimation is started by calling the method `EM_estimate` of the `o_model` object, with the optional `o_EM_options` object passed as an input argument. After model estimation, the variance-covariance matrix of the estimated parameters is evaluated by calling the method `set_varcov`, with the optional approximation level δ of Equation 6 passed as an input parameter. Finally, `set_logL` computes the observed data log-likelihood.

```
o_model.EM_estimate(o_EM_options);
delta = 0.001;
o_model.set_varcov(delta);
o_model.set_logL();
```

All the relevant estimation results are found in the internal '`stem_EM_result`' object, which can be accessed as a property of the `o_model` object as follows:

```
o_model.plot_par;
o_model.beta_Chi2_test;
o_model.print;
```

Figure 4 is produced by calling the `plot_par` method and shows the estimated $\boldsymbol{\beta}_0(h)$, $\boldsymbol{\beta}_{temp}(h)$, $\boldsymbol{\beta}_{uvb}(h)$, and $\sigma_\varepsilon^2(h)$. Thanks to the use of a Fourier basis, the functions are periodic with a period of one day. In the plot of $\sigma_\varepsilon^2(h)$, the unexplained portion of O_3 variance, $\sigma_\varepsilon^2(h)$, is small during daylight hours, which is consistent with the results of Dohan and Masschelein (1987).

When the confidence bands of `parplot` contain zero, it may be useful to test the significance of the covariates. By calling the method `beta_Chi2_test`, the results of χ^2 tests are obtained, and they are reported in Table 1. Although β_{uvb} is close to 0 in the morning, all fixed effects are highly significant overall. The model output is shown in the MATLAB command window by calling the `print` method.

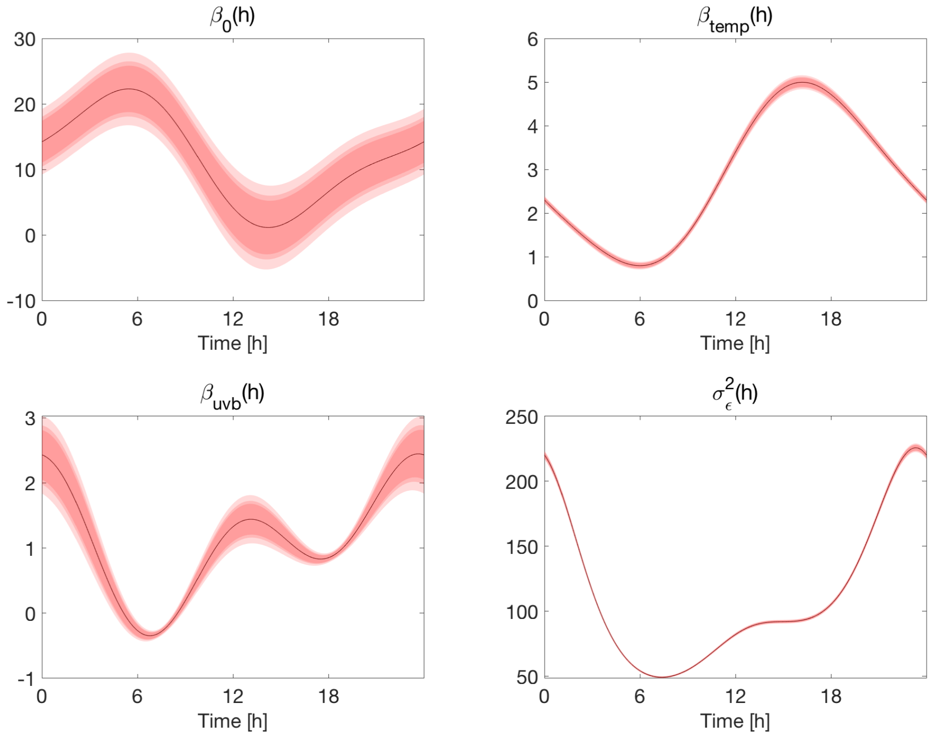

Figure 4: Estimated $\beta_0(h)$, $\beta_{temp}(h)$, $\beta_{uvb}(h)$ and $\sigma^2_\epsilon(h)$, with 90%, 95%, 99% confidence bands, respectively, shown through the different shades.

Validation

This paragraph describes the script `demo_section4_validation.m`, which implements validation. Compared to the code in `demo_section4_model_estimate.m`, it only differs in providing an object of class '`stem_validation`'.

To create the object called `o_validation`, the name of the validation variable is needed as well as the indices of the validation stations. Moreover, if the size of the nearest neighbor set for each kriging site (`nn_size`) is not provided as the third input argument in the '`stem_validation`' class constructor, **D-STEM** v2 uses all the remaining stations. For example, a validation data set with three stations is constructed as follows:

```
S_val = [1, 7, 10];
input_data.stem_validation = stem_validation('O3', S_val);
```

The validation statistics, computed by `EM_estimate`, are saved in the internal object called `stem_validation_result`, which can be accessed as a property of the `o_model` object. The `stem_validation_result` object contains the estimated O_3 residuals for the above-mentioned validation stations as well as the validation mean square errors and R^2, as defined in Section 2.7.

Kriging

This paragraph describes the `demo_section4_kriging.m` script, which applies the approach of Section 2.6 to the estimated model to map the O_3 concentrations over Beijing city.

The first step is to create an object of class 'stem_grid', which collects the information about the regular grid of pixels $\mathcal{B}$ to be used for mapping. Then, an object of class 'stem_krig_data' is created, where the o_krig_grid object is passed as an input argument:

```
load Output/ozone_model;
step_deg = 0.05;
lat = 39.4:step_deg:41.1;
lon = 115.4:step_deg:117.5;
[lon_mat, lat_mat] = meshgrid(lon, lat);
krig_coordinates = [lat_mat(:) lon_mat(:)];
o_krig_grid = stem_grid(krig_coordinates, 'deg', 'regular', 'pixel', ...
  size(lat_mat), 'square', 0.05, 0.05);
o_krig_data = stem_krig_data(o_krig_grid);
```

Two comments on the above lines follow. First, since the grid in the o_krig_grid object is regular, the dimensions of the grid (size(lat_mat), 35 × 43), must be provided as well as the shape of the pixels and the spatial resolution of the grid, which is 0.05° × 0.05°. Second, the above step using the stem_krig_data constructor may appear redundant at first glance. Indeed, it is needed for compatibility with other model types for which, in addition to the 'stem_grid' object, other information is also necessary for the stem_krig_data constructor.

Next, the 'stem_krig_options' class provides some options for kriging. By default, the output is back-transformed in the original unit of measure if the observations have been log-transformed and/or standardized. The back_transform property enables handling this. Moreover, the no_varcov property must be set to 1 to avoid the time-consuming computation of the kriging variance. Eventually, the block_size property is used to define the number of spatial locations in $\mathcal{S}_l^*$.

```
o_krig_options = stem_krig_options();
o_krig_options.back_transform = 0;
o_krig_options.no_varcov = 0;
o_krig_options.block_size = 30;
```

After storing the map of Beijing boundaries into the o_model object, the latter is used with o_krig_data to create an object of class 'stem_krig'. This and o_krig_options together contain all information for kriging, which is obtained by the corresponding kriging method:

```
o_model.stem_data.shape = shaperead('Maps/Beijing_adm1.shp');
o_krig = stem_krig(o_model, o_krig_data);
o_krig_result = o_krig.kriging(o_krig_options);
```

Note that this task may be time consuming for large grids. The kriging output saved in the o_krig_result object gives the latent process estimate z_t and its variance. The surface_plot and profile_plot methods may be used to obtain and plot $\hat{f}(\boldsymbol{s}, h, t)$ of Equation 7. In this case, the user has to provide the corresponding covariate (X_beta) for the scale/vector h, time t or location $\boldsymbol{s}$ (lon0, lat0) of interest.

Specifically, the surface_plot method is used to display the O_3 map using h, t, X_beta as input arguments. In the case of unavailable X_beta, the mapping concerns the component

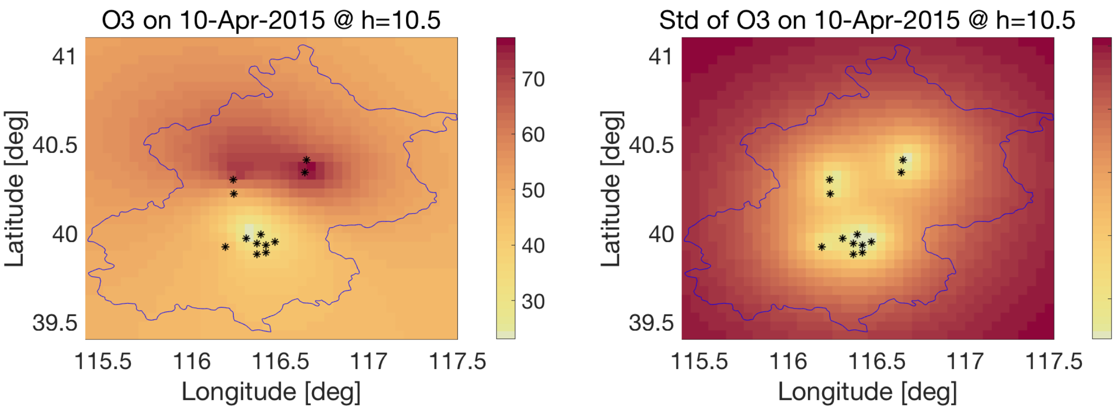

Figure 5: O_3 concentrations and their standard deviation at 10:30 am ($h = 10.5$), on 10 April 2015, where 12 stations are marked with black stars.

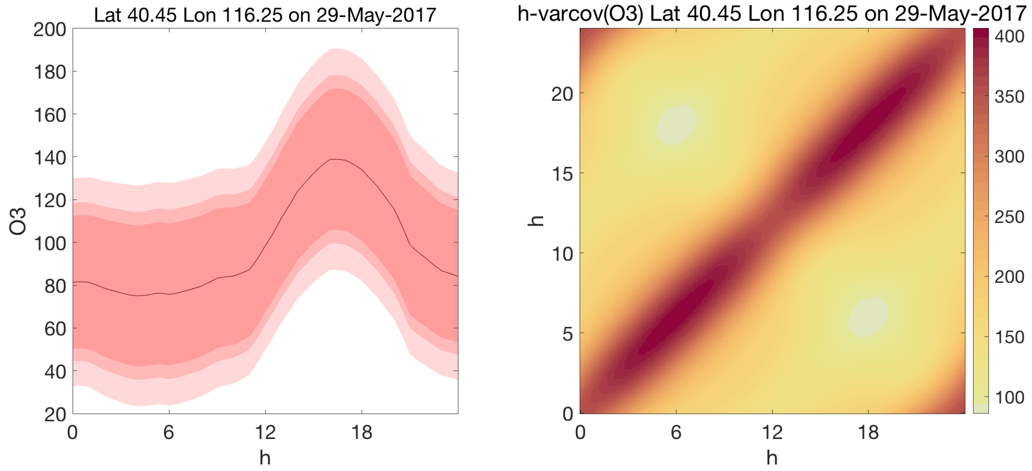

Figure 6: $O3$ concentrations with $90\%, 95\%, 99\%$ confidence bands (different shadings), and their variance-covariance at latitude 40.45, longitude 116.25 on 29 May 2017.

$\boldsymbol{\phi}(h)^\top \boldsymbol{z}(\boldsymbol{s}, t)$. Loaded from the homonym file, the array X_beta_t_100 refers to time $t = 100$ and hour $h = 10.5$ and has the dimension $35 \times 43 \times 3$. Maps of O_3 concentrations and their standard deviation are shown in Figure 5.

```
load Data/kriging/X_beta_t_100;
t = 100;
h = 10.5;
[y_hat, diag_Var_y_hat] = o_krig_result.surface_plot(h, t, X_beta_t_100);
```

On the other hand, the profile_plot method is used to display the O_3 profile at a given spatial location $\boldsymbol{s}$(lon0, lat0) and time t. Still, the profile plot concerns the component $\boldsymbol{\phi}(h)^\top \boldsymbol{z}(\boldsymbol{s}, t)$ if X_beta is not provided. After loading X_beta_h (dimension 25×3) from the homonym file, this method represents the profile of O_3 concentrations and their variance-covariance matrix as in Figure 6:

```
load Data/kriging/X_beta_h;
h = 0:24;
lon0 = 116.25;
lat0 = 40.45;
t = 880;
[y_hat, diag_Var_y_hat] = o_krig_result.profile_plot(h, lon0, lat0, ...
  t, X_beta_h);
```

Note that the prediction in Equation 7 and the variance in Equation 8 are stored in the output arguments `y_hat`, and `diag_Var_y_hat`, respectively.

5. Case study on climate data

In order to show the complexity-reduction capabilities of **D-STEM** v2, a data set of temperature vertical profiles collected by the radiosondes of the Universal Radiosonde Observation Program (RAOB) is now considered. The profiles are observed over the Earth's sphere, and they are misaligned, that is, each profile differs in terms of the number of observations and altitude above the ground of each observation. Additionally, the computation burden is higher due to the higher number of spatial locations at which profiles are observed.

5.1. RAOB data

Radiosondes are routinely launched from stations all over the world to measure the state of the upper troposphere and lower stratosphere. Data collected by radio sounding have applications in weather prediction and climate studies.

Temperature data from 200 globally distributed stations collected daily during January 2015 at 00:00 and 12:00 UTC are considered here. Each profile consists of a given number of measurements taken at different pressure levels. Since the weather balloon carrying the radiosonde usually explodes at an unpredictable altitude, the profile measurements are misaligned across the profiles and have different pressure ranges. A functional data approach is natural in this case since the underlying temperature profile can be seen as a continuous function sampled at some pressure levels.

Figure 1 depicts the spatial locations of temperature measurements taken on 1 January 2015 at 00:00 UTC. This demo data set, which only covers one month, includes around 10^5 data points. When the full data set is used in climate studies, the number of data points grows to around 10^8. In this case, a recent server machine with multiple CPUs with at least 256 GB of RAM is required for model estimation and kriging.

The focus of the case study is on the difference between the radiosonde measurement and the output of the ERA-Interim global atmospheric reanalysis model provided by ECMWF. In particular, the aim is to study the spatial structure of this difference in 4D space, where the dimensions are latitude, longitude, altitude and time.

The model for temperature y is as follows

$$y\left(\boldsymbol{s},h,t\right)=x_{\mathrm{ERA}}\left(\boldsymbol{s},h,t\right)\beta_{\mathrm{ERA}}\left(h\right)+\boldsymbol{\phi}\left(h\right)^{\top}\boldsymbol{z}\left(\boldsymbol{s},t\right)+\varepsilon\left(\boldsymbol{s},h,t\right),$$

where $h\in[50,925]$ hPa is the pressure level, while $t=1,\ldots,62$ is a discrete time index for

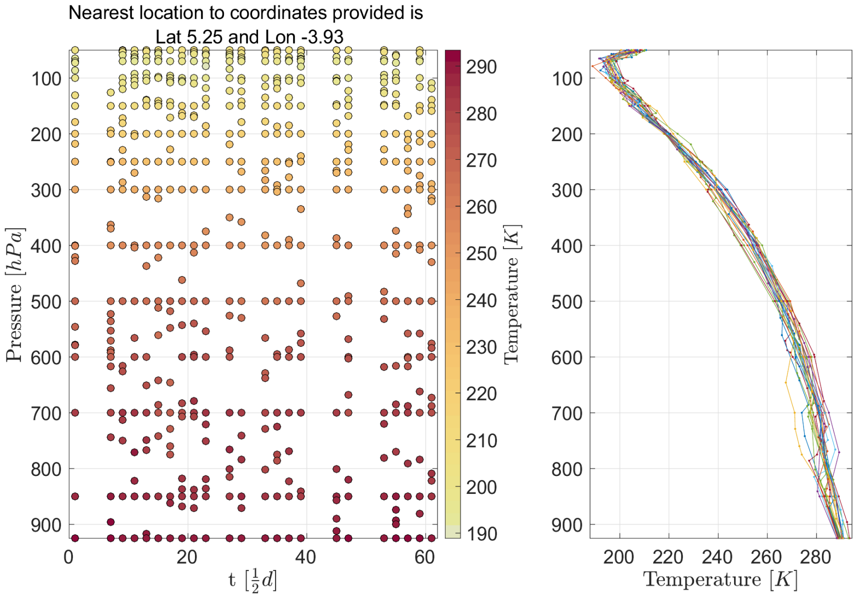

Figure 7: Temperature at location 5.25 latitude and −3.93 longitude in January 2015. Left: each dot is a temperature measurement. The color of the dot depicts the temperature. Right: each graph is a temperature vertical profile collected through radio sounding.

January 2015. Figure 7 shows the temperature measurements at a given station, where 50 and 925 hPa correspond approximately to 25 and 1.3 km, respectively.

5.2. Software implementation

This section details the software implementation of the case study described above as in script `demo_section5.m`, which can be also executed in the `code.m` script. To avoid repetition, only the relevant parts of the script that differ from the case study of Section 4 are reported and commented on here. In particular, data loading and instantiation of the '`stem_model`' object are not described.

Model estimation

The problem of vertical misalignment of the measurements is completely transparent to the user and is handled by the internal `stem_misc.data_formatter` method when creating the '`stem_data`' object. Note that the dimension of the matrices in `o_varset` depends on q, the maximum number of measurements in each profile. To prevent out-of-memory problems, it is advisable to avoid data sets in which only a few profiles have a large number of measurements, which could result in large matrices in `o_varset`, with most of the elements set to `NaN`.

B-spline bases are used, since, in this application, vertical profiles are not periodic with respect to the pressure domain. The corresponding object of class '`stem_fda`' is created in the following way:

```
spline_order = 2;
rng_spline = [50, 925];
knots_number = 5;
```

```
knots = linspace(rng_spline(1), rng_spline(2), knots_number);

input_fda.spline_type = 'Bspline';
input_fda.spline_order = spline_order;
input_fda.spline_knots = knots;
input_fda.spline_range = rng_spline;
o_fda = stem_fda(input_fda);
```

Note that the knots are equally spaced along the functional range. In general, however, non-equally spaced knots can be provided, and each model component (i.e., σ_ε^2, β_j and $\boldsymbol{\phi}(h)^\top \boldsymbol{z}(\boldsymbol{s},t)$) can have a different set of knots. This is obtained using `spline_order` and `spline_knots` with additional suffixes `_sigma`, `_beta`, `_z`.

Although this data set is not large, the demo shows how to enable the partitioning discussed in Section 2.4. First, the spatial locations are partitioned using the modified k-means algorithm:

```
k = 5;
trials = 100;
lambda = 5000;
partitions = o_data.kmeans_partitioning(k, trials, lambda);
```

where `k` is the number of partitions, `trials` is the number of times when the k-means algorithm is executed starting from randomized centroids and `lambda` is λ in Equation 4.

At the end of the k-means algorithm, data are internally re-ordered for parallel computing. Model estimation is done after creating and setting an object of class '`stem_EM_options`'. To do this, the output of the `kmeans_globe` method is passed to the `partitioning_block_size` property of the `o_EM_options` object. Additionally, for parallel computing, the number of workers must be set to a value higher than 1. In general, this could be any number up to the number of cores available on the machine.

```
o_EM_options = stem_EM_options();
o_EM_options.partitions = partitions;
o_EM_options.workers = 2;
o_model.EM_estimate(o_EM_options);
```

The three validation MSEs defined in Section 2.7 are shown in Figures 8 and 9. To generate these figures, the method `plot_validation` is called with vertical = 1, which provides "atmospheric profile" plots with h on the vertical axis:

```
vertical = 1;
o_model.plot_validation(vertical);
```

Kriging

Interpolation across space and over time is done as in Section 4.2. However, complexity reduction is enabled by adopting the nearest neighbor approach detailed in Section 2.6.

To do this, a class constructor is first called, where the `block_size` is used to define the number of spatial locations in $\mathcal{S}_l^*$, and then `nn_size` is used to define $\tilde{n}$. Additionally, setting `o_krig_options.workers` makes it possible to do the kriging over the u blocks in parallel using up to the allocated number of workers:

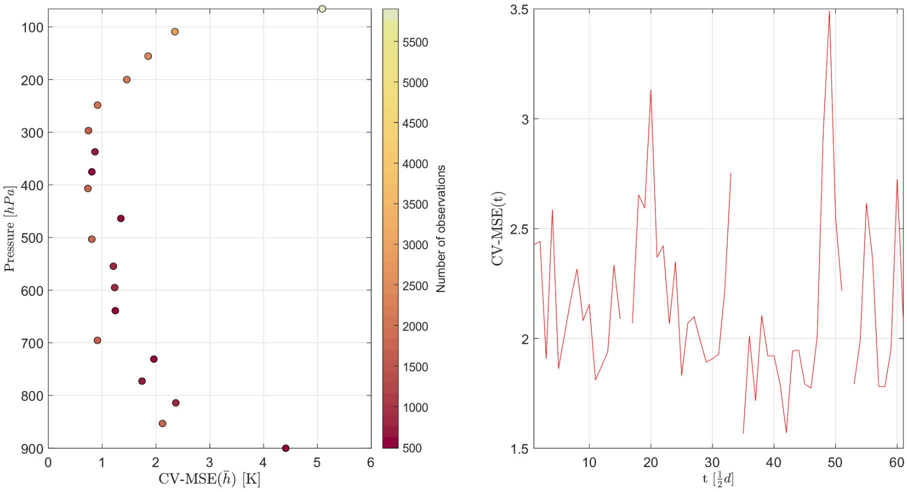

Figure 8: (Left) Validation MSE with respect to the $\bar{h}$ colored by the number of observations n_b; (Right) Validation MSE with respect to time t.

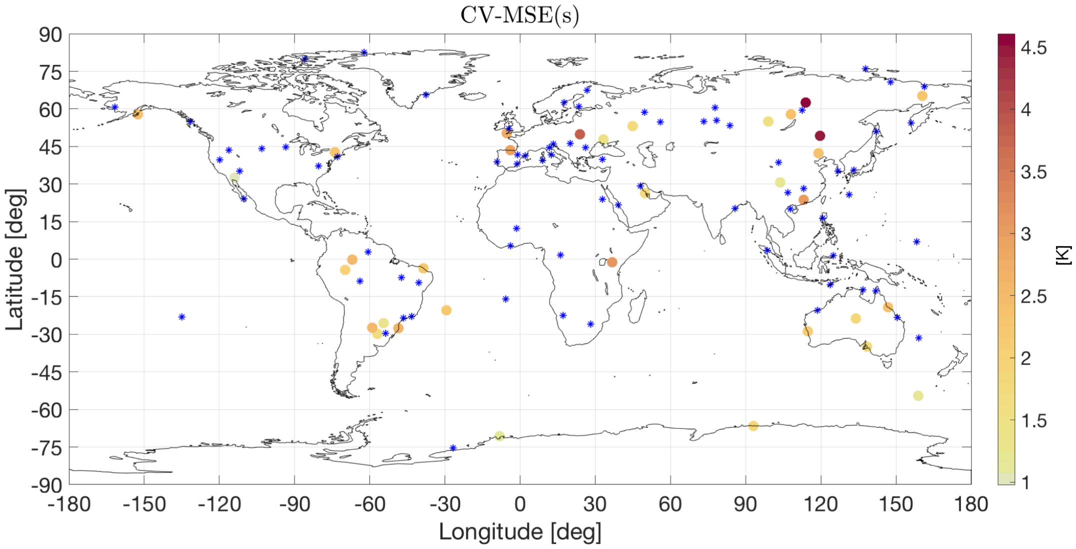

Figure 9: Validation MSE for the thirty-three stations, where the stations used for estimation are marked with blue stars.

```
o_krig_options = stem_krig_options();
o_krig_options.block_size = 150;
o_krig_options.nn_size = 10;
o_krig_options.workers = 2;
```

Finally, kriging predictions and standard errors are mapped for a given $h \in \mathcal{H}$ and time t:

```
h = 875.3;
t = 12;
o_krig_result.surface_plot(h, t);
```

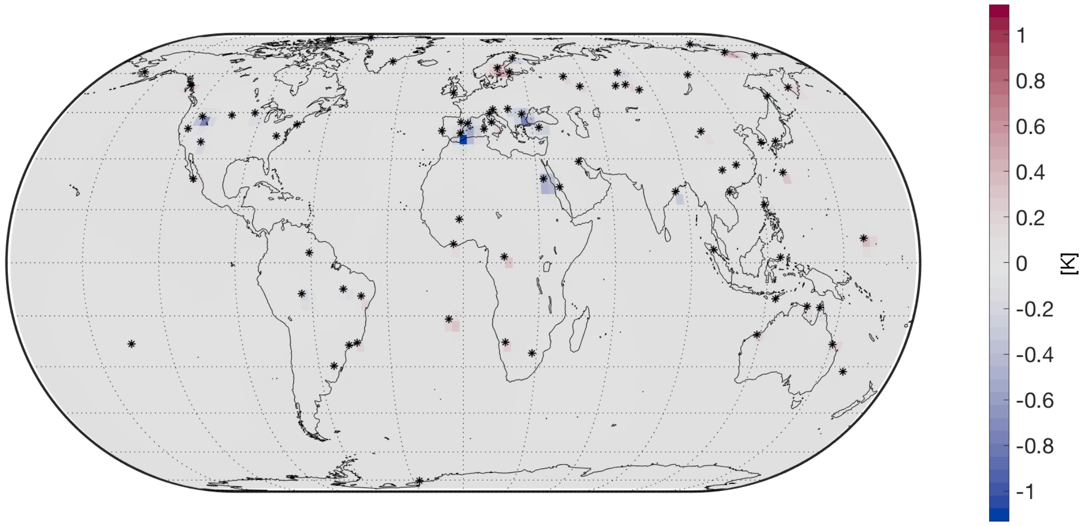

Figure 10: $\boldsymbol{\phi}(h)^\top \hat{\boldsymbol{z}}(\boldsymbol{s},t)$ at pressure 875.3 hPa, and 12:00 am on 6 January 2015, where 200 stations are shown as black stars.

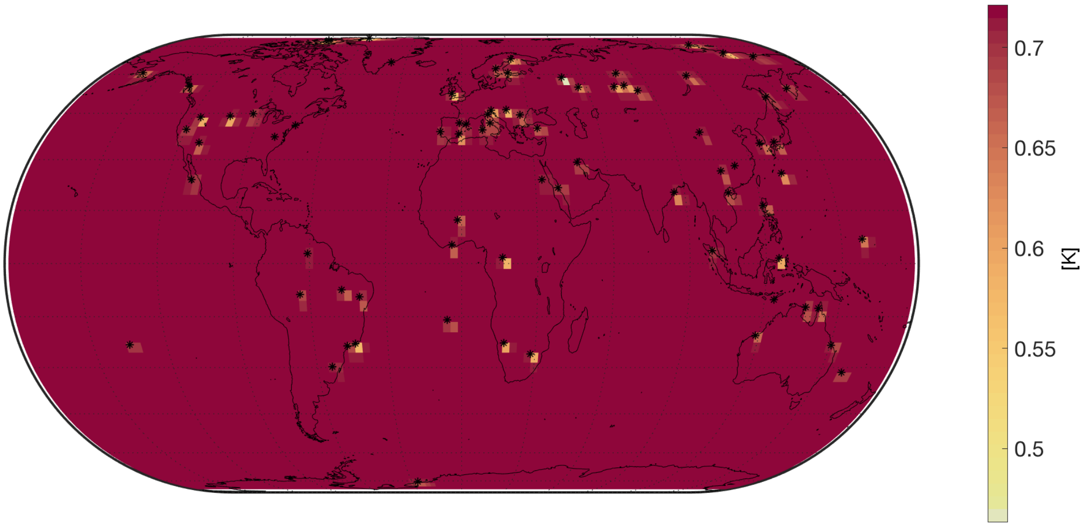

Figure 11: Standard deviation of $\boldsymbol{\phi}(h)^\top \hat{\boldsymbol{z}}(\boldsymbol{s},t)$ at pressure 875.3 hPa, and 12:00 am on 06 January 2015, where 200 stations are shown as black stars.

Since covariates are not provided to the `surface_plot` method, the plots are on the component $\boldsymbol{\phi}(h)^\top \hat{\boldsymbol{z}}(\boldsymbol{s},t)$, namely, the difference between RAOB and ERA-Interim and its standard deviation. The output of the above code is depicted in Figures 10 and 11.

6. Concluding remarks

This paper introduced the package **D-STEM** v2 through two case studies of spatio-temporal modeling of functional data. It is shown that, in addition to maximum likelihood estimation, Hessian approximation and kriging for large data sets, **D-STEM** v2 also develops several data-handling capabilities, allows for automatic construction of relevant objects and provides graphical output. In particular, it provides high-quality global maps and two kinds of functional plotting: the traditional x-y plot and the vertical profile plot, which is popular, for example, in atmospheric data analysis. In this regard, model validation and kriging are straightforward.

D-STEM v2 fills a gap in functional geostatistics. In fact, although statistical methods for georeferenced functional data have been recently developed (e.g., Ignaccolo, Mateu, and Giraldo 2014), standard geostatistical packages do not consider functional data, especially in the spatio-temporal context.

The successful use of **D-STEM** v1 in a number of applications proved that the EM algorithm implementation is quite stable. Now, due to improvements in computational efficiency, the new **D-STEM** v2 has the capability to handle large data sets. Moreover, thanks to the approximated variance-covariance matrix, it is possible to compute standard errors for all model parameters relatively fast and avoid the large number of iterations typically required by an MCMC approach for making inferences.

However, a limit of the EM algorithm is its limited flexibility to changes in the model equations. Indeed, changes in parameterization or latent variable structure usually require deriving new closed-form estimation formulas and changing the software accordingly. Moreover, changes in covariance functions are not easy to handle.

Computationally, the main limit of **D-STEM** v2 is in the number p of basis functions that can be handled. Even if partitioning is exploited in k blocks of size r, computational complexity is $\mathcal{O}\left(Tkr^3p^3\right)$, meaning that p cannot be large.

Currently, the authors are working on a new version, which makes it possible to handle multivariate functional space-time data and user-defined spatial covariance functions, which will make **D-STEM** v2 a valid and comprehensive alternative to the Gaussian process regression models (`fitrgp`) implemented in the Statistics and Machine Learning Toolbox of MATLAB.

Acknowledgments

We would like to thank China's National Key Research Special Program Grant (under the grant No. 2016YFC0207701) and the Center for Statistical Science at Peking University.

References

Bivand RS, Gómez-Rubio V, Rue H (2015). "Spatial Data Analysis with R-**INLA** with Some Extensions." *Journal of Statistical Software*, **63**(20), 1–31. doi:10.18637/jss.v063.i20.

Blangiardo M, Cameletti M, Baio G, Rue H (2013). "Spatial and Spatio-Temporal Models with R-**INLA**." *Spatial and Spatio-Temporal Epidemiology*, **4**, 33–49. doi:10.1016/j.sste.2012.12.001.

Calculli C, Fassò A, Finazzi F, Pollice A, Turnone A (2015). "Maximum Likelihood Estimation of the Multivariate Hidden Dynamic Geostatistical Model with Application to Air Quality in Apulia, Italy." *Environmetrics*, **26**(6), 406–417. doi:10.1002/env.2345.

Cressie N (2018). "Mission CO_2ntrol: A Statistical Scientist's Role in Remote Sensing of Atmospheric Carbon Dioxide." *Journal of the American Statistical Association*, **113**(521), 152–168. doi:10.1080/01621459.2017.1419136.

Cressie N, Johannesson G (2008). "Fixed Rank Kriging for Very Large Spatial Data Sets." *Journal of the Royal Statistical Society B*, **70**(1), 209–226. doi:10.1111/j.1467-9868.2007.00633.x.

Dohan JM, Masschelein WJ (1987). "The Photochemical Generation of Ozone: Present State-of-the-Art." *Ozone: Science & Engineering*, **9**(4), 315–334. doi:10.1080/01919518708552147.

Eaton JW, Bateman D, Hauberg S, Wehbring R (2020). *GNU Octave Version 6.1.0 Manual: A High-Level Interactive Language for Numerical Computations*. URL https://www.gnu.org/software/octave/.

Fassò A (2013). "Statistical Assessment of Air Quality Interventions." *Stochastic Environmental Research and Risk Assessment*, **27**(7), 1651–1660. doi:10.1007/s00477-013-0702-5.

Fassò A, Finazzi F (2011). "Maximum Likelihood Estimation of the Dynamic Coregionalization Model with Heterotopic Data." *Environmetrics*, **22**(6), 735–748. doi:10.1002/env.1123.

Fassò A, Finazzi F, Ndongo F (2016). "European Population Exposure to Airborne Pollutants Based on a Multivariate Spatio-Temporal Model." *Journal of Agricultural, Biological, and Environmental Statistics*, **21**(3), 492–511. doi:10.1007/s13253-016-0260-7.

Fassò A, Ignaccolo R, Madonna F, Demoz B, Franco-Villoria M (2014). "Statistical Modelling of Collocation Uncertainty in Atmospheric Thermodynamic Profiles." *Atmospheric Measurement Techniques*, **7**(6), 1803–1816. doi:10.5194/amt-7-1803-2014.

Finazzi F (2020). "Fulfilling the Information Need after an Earthquake: Statistical Modelling of Citizen Science Seismic Reports for Predicting Earthquake Parameters in Near Realtime." *Journal of the Royal Statistical Society A*, **183**(3), 857–882. doi:10.1111/rssa.12577.

Finazzi F, Fassò A (2014). "**D-STEM**: A Software for the Analysis and Mapping of Environmental Space-Time Variables." *Journal of Statistical Software*, **62**(6), 1–29. doi:10.18637/jss.v062.i06.

Finazzi F, Fassò A, Madonna F, Negri I, Sun B, Rosoldi M (2019a). "Statistical Harmonization and Uncertainty Assessment in the Comparison of Satellite and Radiosonde Climate Variables." *Environmentrics*, **30**(2), e2528. doi:10.1002/env.2528.

Finazzi F, Haggarty R, Miller C, Scott M, Fassò A (2015). "A Comparison of Clustering Approaches for the Study of the Temporal Coherence of Multiple Time Series." *Stochastic Environmental Research and Risk Assessment*, **29**(2), 463–475. doi:10.1007/s00477-014-0931-2.

Finazzi F, Napier Y, Scott M, Hills A, Cameletti M (2019b). "A Statistical Emulator for Multivariate Model Outputs with Missing Values." *Atmospheric Environment*, **199**, 415–422. doi:10.1016/j.atmosenv.2018.11.025.

Finazzi F, Scott EM, Fassò A (2013). "A Model-Based Framework for Air Quality Indices and Population Risk Evaluation, with an Application to the Analysis of Scottish Air Quality Data." *Journal of the Royal Statistical Society C*, **62**(2), 287–308. doi:10.1111/rssc.12001.

Finazzi F, Wang Y, Fassó A (2021). ***D-STEM**: Distributed Space Time Expecation Maximization*. MATLAB package v2, URL https://github.com/graspa-group/d-stem.

Finley A, Banerjee S, Gelfand A (2015). "**spBayes** for Large Univariate and Multivariate Point-Referenced Spatio-Temporal Data Models." *Journal of Statistical Software*, **63**(13), 1–28. doi:10.18637/jss.v063.i13.

Furrer R, Genton MG, Nychka D (2006). "Covariance Tapering for Interpolation of Large Spatial Datasets." *Journal of Computational and Graphical Statistics*, **15**(3), 502–523. doi:10.1198/106186006x132178.

Gasch CK, Hengl T, Gräler B, Meyer H, Magney TS, Brown DJ (2015). "Spatio-Temporal Interpolation of Soil Water, Temperature, and Electrical Conductivity in 3D+T: The Cook Agronomy Farm Data Set." *Spatial Statistics*, **14**(A), 70–90. doi:10.1016/j.spasta.2015.04.001.

Gramacy RB (2016). "**laGP**: Large-Scale Spatial Modeling via Local Approximate Gaussian Processes in R." *Journal of Statistical Software*, **72**(1), 1–46. doi:10.18637/jss.v072.i01.

Heaton MJ, Datta A, Finley AO, Furrer R, Guinness J, Guhaniyogi R, Gerber F, Gramacy RB, Hammerling D, Katzfuss M, Lindgren F, Nychka DW, Sun F, Zammit-Mangion A (2019). "A Case Study Competition among Methods for Analyzing Large Spatial Data." *Journal of Agricultural, Biological and Environmental Statistics*, **24**(3), 398–425. doi:10.1007/s13253-018-00348-w.

Ignaccolo R, Mateu J, Giraldo R (2014). "Kriging with External Drift for Functional Data for Air Quality Monitoring." *Stochastic Environmental Research and Risk Assessment*, **28**(5), 1171–1186. doi:10.1007/s00477-013-0806-y.

Jurek M, Katzfuss M (2021). "Multi-Resolution Filters for Massive Spatio-Temporal Data." *Journal of Computational and Graphical Statistics*. doi:10.1080/10618600.2021.1886938. Forthcoming.

Kahle D, Wickham H (2013). "**ggmap**: Spatial Visualization with **ggplot2**." *The R Journal*, **5**(1), 144–161. doi:10.32614/rj-2013-014.

Katzfuss M (2017). "A Multi-Resolution Approximation for Massive Spatial Datasets." *Journal of the American Statistical Association*, **112**(517), 201–214. doi:10.1080/01621459.2015.1123632.

Lindgren F, Rue H (2015). "Bayesian Spatial Modelling with R-**INLA**." *Journal of Statistical Software*, **63**(19), 1–25. doi:10.18637/jss.v063.i19.

Negri I, Fassò A, Mona L, Papagiannopoulos N, Madonna F (2018). "Modeling Spatiotemporal Mismatch for Aerosol Profiles." In *Quantitative Methods in Environmental and Climate Research*, pp. 63–83. Springer-Verlag.

Nychka D, Bandyopadhyay S, Hammerling D, Lindgren F, Sain S (2015). "A Multiresolution Gaussian Process Model for the Analysis of Large Spatial Datasets." *Journal of Computational and Graphical Statistics*, **24**(2), 579–599. doi:10.1080/10618600.2014.914946.

Nychka D, Hammerling D, Sain S, Lenssen N (2019). ***LatticeKrig**: Multiresolution Kriging Based on Markov Random Fields.* University Corporation for Atmospheric Research, Boulder. doi:10.5065/D6HD7T1R. R package version 8.4, URL https://CRAN.R-project.org/package=LatticeKrig.

Pebesma E (2012). "**spacetime**: Spatio-Temporal Data in R." *Journal of Statistical Software*, **51**(7), 1–30. doi:10.18637/jss.v051.i07.

Pebesma E, Heuvelink G (2016). "Spatio-Temporal Interpolation Using **gstat**." *The R Journal*, **8**(1), 204–218. doi:10.32614/rj-2016-014.

Porcu E, Alegria A, Furrer R (2018). "Modeling Temporally Evolving and Spatially Globally Dependent Data." *International Statistical Review*, **86**(2), 344–377. doi:10.1111/insr.12266.

Ramsay JO (2020). ***fda**: Functional Data Analysis.* R package version 5.1.9, URL https://CRAN.R-project.org/package=fda.

Ramsay JO, Silverman BW (2007). *Applied Functional Data Analysis: Methods and Case Studies*. Springer-Verlag. doi:10.1007/b98886.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rue H, Martino S, Blangiardo FL, Simpson D, Riebler A, Krainski ET (2014). ***INLA**: Functions Which Allow to Perform Full Bayesian Analysis of Latent Gaussian Models Using Integrated Nested Laplace Approximaxion.* R package version 0.0-1404466487, URL http://www.R-INLA.org/.

Shumway RH, Stoffer DS (2017). *Time Series Analysis and Its Applications: With R Examples*. Springer-Verlag.

Stein ML (2002). "The Screening Effect in Kriging." *The Annals of Statistics*, **30**(1), 298–323. doi:10.1214/aos/1015362194.

Stein ML (2013). "Statistical Properties of Covariance Tapers." *Journal of Computational and Graphical Statistics*, **22**(4), 866–885. doi:10.1080/10618600.2012.719844.

Taghavi-Shahri SM, Fassò A, Mahaki B, Amin H (2019). "Concurrent Spatiotemporal Daily Land Use Regression Modeling and Missing Data Imputation of Fine Particulate Matter Using Distributed Space-Time Expectation Maximization." *Atmospheric Environment*, **224**, 117202. doi:10.1016/j.atmosenv.2019.117202.

The MathWorks Inc (2021). *MATLAB – The Language of Technical Computing, Version R2021a*. Natick, Massachusetts. URL http://www.mathworks.com/products/matlab/.

Tzeng S, Huang HC (2018). "Resolution Adaptive Fixed Rank Kriging." *Technometrics*, **60**(2), 198–208. doi:10.1080/00401706.2017.1345701.

Wan Y, Xu M, Huang H, Chen S (2021). "A Spatio-Temporal Model for the Analysis and Prediction of Fine Particulate Matter Concentration in Beijing." *Environmetrics*, **32**, e2648. doi:10.1002/env.2648.

Wang T, Xue L, Brimblecombe P, Lam YF, Li L, Zhang L (2017). "Ozone Pollution in China: A Review of Concentrations, Meteorological Influences, Chemical Precursors, and Effects." *Science of the Total Environment*, **575**, 1582–1596. doi:10.1016/j.scitotenv.2016.10.081.

Zammit-Mangion A, Cressie N (2021). "**FRK**: An R Package for Spatial and Spatio-Temporal Prediction with Large Datasets." *Journal of Statistical Software*, **98**(4), 1–48. doi:10.18637/jss.v098.i04.

Zammit-Mangion A, Sainsbury-Dale M (2021). ***FRK**: Fixed Rank Kriging.* R package version 2.0.0, URL https://CRAN.R-project.org/package=FRK.

Affiliation:

Yaqiong Wang
Guanghua School of Management
Peking University
Yiheyuan road, 5
100871 Peking, China
E-mail: yaqiongwang@pku.edu.cn

Francesco Finazzi, Alessandro Fassò
Department of Management, Information and Production Engineering
University of Bergamo
viale Marconi, 5
24044 Dalmine (BG), Italy
E-mail: francesco.finazzi@unibg.it, alessandro.fasso@unibg.it

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
September 2021, Volume 99, Issue 10 *Submitted:* 2019-07-02
doi:10.18637/jss.v099.i10 *Accepted:* 2020-11-15