Journal of Statistical Software

December 2014, Volume 62, Issue 6. *http://www.jstatsoft.org/*

D-STEM: A Software for the Analysis and Mapping of Environmental Space-Time Variables

Francesco Finazzi
University of Bergamo

Alessandro Fassò
University of Bergamo

Abstract

This paper discusses the software **D-STEM** as a statistical tool for the analysis and mapping of environmental space-time variables. The software is based on a flexible hierarchical space-time model which is able to deal with multiple variables, heterogeneous spatial supports, heterogeneous sampling networks and missing data. Model estimation is based on the expectation maximization algorithm and it can be performed using a distributed computing environment to reduce computing time when dealing with large data sets. The estimated model is eventually used to dynamically map the variables over the geographic region of interest. Three examples of increasing complexity illustrate usage and capabilities of **D-STEM**, both in terms of modeling and implementation, starting from a univariate model and arriving at a multivariate data fusion with tapering.

Keywords: multivariate space-time models, data fusion, remote sensing, expectation maximization, MATLAB.

1. Introduction

The understanding of complex environmental phenomena usually requires the analysis of multiple variables observed over space and time, resulting in possibly large and complex data sets. When multivariate space-time data sets are considered, it is common to rely on statistical spatio-temporal models able to exploit the correlation across variables and to provide space-time predictions over the geographic region of interest (Cressie and Wikle 2011).

This paper introduces the **D-STEM** (distributed space time expectation maximization) software as a statistical tool for the analysis of environmental space-time data sets and the prediction, uncertainty included, of the observed variables.

D-STEM is developed in the MATLAB (The MathWorks, Inc. 2010) language and it is available at `https://code.google.com/p/d-stem/`. The modeling capabilities of **D-STEM** are

detailed in this paper by introducing three case studies of increasing complexity. The reader can download the `D-STEM_v4.7.11_Full.zip` archive – either from the journal web page or from the link above – including source code and demo folders with replication materials. Please follow the instructions given in the `ReadMe.txt` file of the demo folder to reproduce the case studies. **D-STEM** requires the **Statistics** toolbox, the **Optimization** toolbox and the **Mapping** toolbox (The MathWorks, Inc. 2010).

D-STEM is the evolution of the R (R Core Team 2014) package **Stem** (Cameletti 2012) which provides space-time data modeling capabilities by means of hierarchical space-time models within the frequentist paradigm. Excluding the many packages for spatial data, only few R packages can handle space-time data and even fewer are suitable for multivariate space-time data. The R package **spTimer** (Bakar and Sahu 2014b,a) implements space-time models similar to those implemented by **Stem** but model estimation is performed within the Bayesian setting. Compared to the packages **Stem** and **spTimer**, **D-STEM** allows to estimate a larger class of univariate and multivariate hierarchical space-time models and it is optimized for large data sets. The **gstat** package (Pebesma and Gaeler 2013) can deal with multivariate space-time data but data interpolation is based on variogram modeling. When modeling environmental space-time variables, **D-STEM** is an alternative to the R package **INLA** (Rue, Martino, Lindgren, Simpson, Riebler, and Krainski 2014) which implements the integrated nested Laplace approximation (INLA) and the stochastic partial differential equation (SPDE) modeling approaches. Although **D-STEM** and the **INLA** package are based on hierarchical models and latent variables, the space-time models they implement overlap only partially and the user may benefit from using them both depending on the specific application (Cameletti, Lindgren, Simpson, and Rue 2013).

D-STEM has been tested by the authors in various real-data applications. At the urban scale, it has been used for assessing the space-time impact of traffic policies in Milan city (Fassò 2013). At the country scale, it has been used for evaluating multi-variable air quality indexes and for assessing the airborne pollutant exposure distribution in Scotland (Finazzi, Scott, and Fassò 2013). At the continental scale, considering a large data set of both ground level and remote sensing data, it has been used for air quality dynamic mapping over Europe (Fassò and Finazzi 2013).

The rest of the paper is organized as follows. Section 2 describes the capabilities of the software in terms of data modeling and data handling in general terms. Section 3 introduces the software classes at the basis of **D-STEM**. Sections 4, 5 and 6 illustrate software usage and capabilities considering the three case studies implemented in the above mentioned demo, which are based, respectively, on univariate, multivariate and data fusion models. Section 7 describes three options for handling large data sets and in particular tapering, distributed computing and software configuration to reduce the computational burden. Conclusions are given in Section 8.

2. Software description

2.1. Modeling capabilities

The parametric statistical model implemented in **D-STEM** is based on latent space-time random variables and space-time varying coefficients. The varying coefficients can be either

observed covariates or the loadings derived from some basis functions. The model, thus, can reach a high level of flexibility and it is suitable for modeling variables over large geographic regions.

The latent spatial random variables are modeled as Gaussian random fields with a Matérn correlation function and, in the case of multiple variables, the spatial cross-correlation is modeled through the linear coregionalization model (LCM). On the other hand, time is assumed to be discrete and it is modeled through latent temporal random variables with Markovian dynamics.

In many applications, the observations of a variable must be calibrated using the observations of a second variable or a given variable is observed using more than one instrument and/or technique. For instance, remote sensing data are often calibrated using ground level data. **D-STEM** allows to jointly solve the calibration and the data fusion problems. In particular, point data and pixel/block data can be handled in a multivariate setting.

Model parameters are estimated following the maximum likelihood approach by means of the expectation maximization (EM) algorithm. When large data sets are considered, the tapering approach can be used in order to obtain sparse variance-covariance matrices reducing the computing time. If a computer cluster is available, model estimation can be performed in a distributed manner exploiting all the available CPU as well as CPU cores.

Details on the mathematical structure of the model at the basis of **D-STEM** are given in the following sections while model estimation formulas and their derivation can be found in Fassò and Finazzi (2013) and references therein.

2.2. Data handling

Multivariate space-time data sets are challenging as, in general, each variable can be observed at different spatial locations and missing data are the rule rather than the exception. **D-STEM** is able to handle heterotopic data sets where each variable is observed at possibly different sets of spatial locations. The sets of spatial locations or the grids of pixels are assumed to be time invariant. As a consequence, the single observation is considered to be missing if it is not observed at a given time step. Missing data, however, are automatically handled without the need of data imputation or interpolation.

2.3. Model output

The result of model estimation consists of the values of the estimated parameters, their variance-covariance matrix and the observed data log-likelihood. Moreover, cross-validation mean squared error can be obtained for each variable following a 2-fold cross-validation approach. The estimated model is eventually used to dynamically map each variable at high spatial resolution over the geographic region.

3. Software structure

D-STEM is based on the object oriented paradigm. Data handling and analysis are thus performed by creating objects from the **D-STEM** classes and by calling the appropriate methods. The following list describes the classes that the end user should manage.

- Data handling
  - '`stem_varset`' – the class contains the observed data of all the variables and the loading coefficients;
  - '`stem_grid`' – the class contains all the information related to the sampling locations of a single variable;
  - '`stem_gridlist`' – the class is the collector of the `stem_grid` objects for all the variables;
  - '`stem_datestamp`' – the class contains the information related to the date and time of the observations;
  - '`stem_data`' – the class is the collector of the '`stem_varset`', '`stem_gridlist`' and '`stem_datestamp`' objects and it provides methods for preliminary data manipulation.
- Model and model estimation
  - '`stem_par`' – the class contains the structure and the values of the model parameters;
  - '`stem_model`' – the class is the collector of the '`stem_data`' and the '`stem_par`' objects and it provides methods for model estimation;
  - '`stem_EM_options`' – the class includes the options of the EM algorithm used for model estimation;
  - '`stem_crossval`' – the class contains the information needed for cross-validation and the cross-validation result;
  - '`stem_sim`' – the class is used to simulate a data set from a given model.
- Model estimation result
  - '`stem_EM_result`' – the class contains the result of the EM estimation;
  - '`stem_kalmansmoother_result`' – the class contains the output of the Kalman smoother implemented within the EM algorithm.
- Kriging
  - '`stem_krig`' – the class includes all the information needed for mapping a variable over space and time using a dynamic kriging technique;
  - '`stem_krig_result`' – the class contains the result of kriging.
- Auxiliary
  - '`stem_misc`' – the class provides miscellaneous methods used by the mother classes. All the methods of the class '`stem_misc`' are static which implies that they can be called without creating an object of class '`stem_misc`'.

Details on all class constructors, properties and input/output arguments can be displayed using the command `doc <class_name>` in the MATLAB environment.

4. Univariate model

The first case study concerns mapping the concentration of daily average nitrogen dioxide (NO_2) over Northern Italy for 2009, which is measured by $n_1 = 194$ ground level monitoring stations irregularly located over the geographic region. Three covariates are considered: wind speed, $x_{wind}(\mathbf{s}, t)$, which is a space-time covariate, land elevation, $x_{land}(\mathbf{s})$, which is a purely spatial covariate and the dummy $x_{sun}(t)$ for Sunday, which is purely temporal.

The following model for the response variable $y_{NO_2}(\mathbf{s}, t)$ observed at spatial location $\mathbf{s}$ and time t is considered:

$$y_{NO_2}(\mathbf{s}, t) = x_{wind}(\mathbf{s}, t)\beta_1 + x_{land}(\mathbf{s})\beta_2(\mathbf{s}, t) + x_{sun}(t)\beta_3 + z(t) + \varepsilon(\mathbf{s}, t), \tag{1}$$

where $\beta_j, j = 1, \ldots, 3$ are coefficients to be estimated, while $z(t)$ is a stochastic time trend. Note that, $x_{land}(\mathbf{s})$ has a stochastic varying coefficient $\beta_2(\mathbf{s}, t) = \beta_2 + \alpha w(\mathbf{s}, t)$ where β_2 is the global effect of land elevation on NO_2 while $\alpha w(\mathbf{s}, t)$ is the random "variation" of β_2 specific to location $\mathbf{s}$ and time t. Since $w(\mathbf{s}, t)$ has unit variance, α is a scale parameter that can be tested to assess whether $\beta_2(\mathbf{s}, t)$ is constant or not. Finally, $z(t)$ is a scalar Markovian process modeling the temporal persistence of the pollutant while $\varepsilon(\mathbf{s}, t)$ is the measurement error.

4.1. Model description

The model in Equation 1 is a special case of the following general univariate model implemented in **D-STEM**:

$$y(\mathbf{s}, t) = \mu(\mathbf{s}, t) + \omega(\mathbf{s}, t) + \varepsilon(\mathbf{s}, t), \tag{2}$$

where $y(\mathbf{s}, t)$ is the scalar observation at time $t \in \{1, \ldots, T\}$ and spatial location $\mathbf{s} \in \mathcal{D}$. Depending on the coordinate system of the data, two options are available, namely $\mathcal{D} \subset \Re^2$ or $\mathcal{D} \subset S^2$, where S^2 is the sphere in $\Re^3$.

In Equation 2, $\mu(\mathbf{s}, t)$ represents the following fixed effect model:

$$\mu(\mathbf{s}, t) = \mathbf{x}_{\boldsymbol{\beta}}(\mathbf{s}, t)\boldsymbol{\beta}, \tag{3}$$

where $\mathbf{x}_{\boldsymbol{\beta}}(\mathbf{s}, t)$ is a $1 \times b$ dimensional vector of known coefficients and $\boldsymbol{\beta}$ is to be estimated. Moreover $\omega(\mathbf{s}, t)$ represents the following random effect model:

$$\omega(\mathbf{s}, t) = \sum_{j=1}^{c} \alpha_j x_j(\mathbf{s}, t) w_j(\mathbf{s}, t) + \mathbf{x}_{\mathbf{z}}(\mathbf{s}, t)\mathbf{z}(t), \tag{4}$$

where $x_j(\mathbf{s}, t), j = 1, \ldots, c$, and $\mathbf{x}_{\mathbf{z}}(\mathbf{s}, t)$ are scalars and a $1 \times p$ dimensional vector of known coefficients, respectively, while $\alpha_j, j = 1, \ldots, c$, are to be estimated.

The p-dimensional latent component $\mathbf{z}(t)$ has the following Markovian dynamics:

$$\mathbf{z}(t) = \mathbf{G}\mathbf{z}(t-1) + \boldsymbol{\eta}(t)$$

with transition matrix $\mathbf{G}$ assumed to have eigenvalues smaller than 1 in absolute value and innovations $\boldsymbol{\eta}(t) \sim N_p(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\eta}})$. Eventually the variables $w_j(\mathbf{s}, t)$ are zero-mean and unit-variance independent Gaussian processes uncorrelated over time but correlated over space with Matérn spatial covariance function

$$\rho(\|\mathbf{s} - \mathbf{s}'\|; \theta_j, \nu) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\sqrt{2\nu}\frac{\|\mathbf{s} - \mathbf{s}'\|}{\theta_j}\right)^{\nu} K_\nu \left(\sqrt{2\nu}\frac{\|\mathbf{s} - \mathbf{s}'\|}{\theta_j}\right), \tag{5}$$

where $\|\mathbf{s}-\mathbf{s}'\|$ is the distance between two generic spatial locations, $\theta_j > 0$ and $\nu > 0$ are parameters, Γ is the gamma function and K_ν is the modified Bessel function of the second kind. Finally, $\varepsilon(\mathbf{s},t)$ is a zero-mean Gaussian process uncorrelated over space and time with variance σ^2. For this model setup, the parameter set to be estimated is

$$\boldsymbol{\psi} = \{\boldsymbol{\beta}, \sigma^2, \boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{G}, \mathbf{v}_{\boldsymbol{\eta}}\},$$

where $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_c\}$, $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_c\}$ and $\mathbf{v}_{\boldsymbol{\eta}}$ is the $p(p+1)/2$ dimensional vector of unique elements of $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$. Note that, for each Matérn correlation function used, only the parameter θ_j is estimated while the smoothing parameter ν is fixed and can be chosen as $1/2$, $3/2$ or $5/2$.

The model in Equation 2 extends the model developed in Fassò and Finazzi (2011) by allowing the interaction between the latent spatial variables $w_j(\mathbf{s},t)$ and the loading coefficients $\mathbf{x}(\mathbf{s},t)$ and by allowing $y(\mathbf{s},t)$ to be missing. The structure of the model in Equation 2 is quite general and special cases thereof have already been used. For instance, Katzfuss and Cressie (2011) consider a similar model that includes both covariates and loading coefficients from basis functions. Although the software considers space-time varying $x_j(\mathbf{s},t)$, and allows to implement space-time varying coefficients such as the $\beta_2(\mathbf{s},t)$ in our case study, the simpler setup given by $x_j(\mathbf{s},t) = 1$ is quite common to model a spatial trend, see for example Fassò (2013) where spatial correlation is considered a nuisance parameter. In general, we suggest to use coefficients $x_j(\mathbf{s},t)$ which are fixed in space and/or time. This is because $w_j(\mathbf{s},t)$ is itself space-time variant and identifiability issues may occur.

In our case study, the vector $\mathbf{x}_{\boldsymbol{\beta}}(\mathbf{s},t)$ includes all the covariates in Equation 1, $\mathbf{x}_{\mathbf{z}}(\mathbf{s},t) = 1$ while $x_1(\mathbf{s})$ is equal to the land elevation. Moreover, $\nu = 1/2$, namely the exponential correlation function is considered.

4.2. Software implementation

This paragraph describes the relevant lines of code of the `demo_section4.m` script related to the case study previously introduced. The script can be executed choosing option number one from the `dstem_demo.m` script.

It is assumed that observations and covariates are stored in MATLAB format files. In general, the user has to take care of loading the data from external sources and formatting them as requested by the class constructors.

Although not mandatory, the temporary data structure `ground` will be used to pass the data to the class constructors. In the following lines of code, data related to the NO_2 concentration are loaded into the structure `ground` along with the variable name. Note that the term "ground", referring to the monitoring network data, is used to contrast them with "remote" sensing data of Section 6.

```
>> load ../Data/no2_ground.mat
>> ground.Y{1} = no2_ground.data;
>> ground.Y_name{1} = 'no2 ground';
>> n1 = size(ground.Y{1}, 1);
>> T = size(ground.Y{1}, 2);
```

The `no2_ground.data` variable is a $n_1 \times T$ matrix, which is allowed to include `NaN` values for the missing data, where n_1 is the total number of sampling locations and T is the total number of time steps.

Similarly, the loading coefficients are loaded into the same `ground` structure. Note that all the loading coefficients are supposed to be observed without error and/or missing data for each day and each spatial location. The loading coefficients related to $\boldsymbol{\beta}$ are directly obtained from the covariates as detailed below.

```
>> load ../Data/no2_ground_covariates.mat
>> ground.X_beta{1} = X;
>> ground.X_beta_name{1} = {'wind speed', 'elevation', 'sunday'};
```

The variable `X` is a $n_1 \times b \times T$ array, with $b = 3$ the number of loading coefficients. Note that, since wind speed is time-variant, the other covariates are replicated T times in order to fill the third array dimension. If all the loading coefficients are time-invariant, however, `X` is simply a $n_1 \times b$ matrix.

The loading coefficients related to $z(t)$ are constant and equal to 1 and they are defined in the following way.

```
>> ground.X_z{1} = ones(n1, 1);
>> ground.X_z_name{1} = {'constant'};
```

Finally, the loading coefficients $x_1(\mathbf{s}, t)$ for the latent spatial variable $w_1(\mathbf{s}, t)$ are extracted from `ground.X_beta{1}` as it corresponds to the land elevation covariate.

```
>> ground.X_p{1} = ground.X_beta{1}(:, 2, 1);
>> ground.X_p_name{1} = {'elevation'};
```

The suffix "`_p`" in `X_p` and `X_p_name` refers to ground data, which are assumed to be point data, and it is necessary to differentiate them from the remote data of Section 6, which are assumed to be block or pixel data. In general, `X_p` is a $n_1 \times 1 \times T \times c$ array. Since, in this case study, $c = 1$ and land elevation is time invariant, then `X_p` is simply a $n_1 \times 1$ vector.

At this point, the `obj_stem_varset_p` object of class '`stem_varset`' can be created using the class constructor as follows.

```
>> obj_stem_varset_p = stem_varset(ground.Y, ground.Y_name, [], [], ...
     ground.X_beta, ground.X_beta_name, ground.X_z, ground.X_z_name, ...
     ground.X_p, ground.X_p_name);
```

The empty input arguments relate to the pixel data which, in this case study, are not considered.

The next step is to create an object of class '`stem_grid`' and to add it in the following way to the `obj_stem_gridlist_p` object of class '`stem_gridlist`'.

```
>> obj_stem_gridlist_p = stem_gridlist();
>> ground.coordinates{1} = [no2_ground.lat, no2_ground.lon];
>> obj_stem_grid = stem_grid(ground.coordinates{1}, 'deg', 'sparse', ...
     'point');
>> obj_stem_gridlist_p.add(obj_stem_grid);
```

The constructor of the class '`stem_grid`' requires to specify some information about the grid and in particular the unit of measure (degrees, kilometers or meters), the configuration of the spatial locations (sparse or regular) and the grid type (points or pixels).

The temporal information of the observed data is provided as follows and it is used when model output is displayed.

```
>> obj_stem_datestamp = stem_datestamp('01-01-2009 00:00', ...
     '31-12-2009 00:00', T);
```

Note that both date and time must be provided regardless of the temporal granularity of the data (hourly, daily, etc.).

The objects so far created are necessary for the constructor of the class '`stem_data`' to produce the `obj_stem_data` object as follows.

```
>> shape = shaperead('../Maps/worldmap');
>> obj_stem_data = stem_data(obj_stem_varset_p, obj_stem_gridlist_p, [], ...
     [], obj_stem_datestamp, shape);
```

The third and fourth input arguments are empty as they are not required for this case study and will be discussed in Section 6. A custom map of the geographic region can be loaded from a shape file and passed as input argument to the constructor. A map of the world country boundaries is provided along with the case study data.

Along with the information about the type of the spatial correlation function (exponential in this case), the `obj_stem_data` object is needed to create the `obj_stem_par` object of class '`stem_par`'.

```
>> obj_stem_par = stem_par(obj_stem_data, 'exponential');
```

Finally, the `obj_stem_data` and `obj_stem_par` objects are used to create the `obj_stem_model` object of class '`stem_model`'.

```
>> obj_stem_model = stem_model(obj_stem_data, obj_stem_par);
```

In order to improve the numerical stability of the model estimation algorithm, observations and loading coefficients are standardized using the `standardize` method of the class '`stem_data`' as follows.

```
>> obj_stem_model.stem_data.log_transform;
>> obj_stem_model.stem_data.standardize;
```

The `log_transform` method only acts on the response variable $y(\mathbf{s}, t)$ and it is used here to reduce the distribution asymmetry.

The EM algorithm requires the model parameters to be initialized to some starting values. The estimation result may depend on the starting values and they must be chosen carefully. In its current version, **D-STEM** can automatically provide starting values only for the $\boldsymbol{\beta}$ parameter vector. The following lines of code describe the initialization of the model parameters.

```
>> obj_stem_par.beta = obj_stem_model.get_beta0();
>> obj_stem_par.alpha_p = 0.6;
>> obj_stem_par.theta_p = 100;
>> obj_stem_par.v_p = 1;
>> obj_stem_par.sigma_eta = 0.2;
>> obj_stem_par.G = 0.8;
>> obj_stem_par.sigma_eps = 0.3;
>> obj_stem_model.set_initial_values(obj_stem_par);
```

Note that the `theta_p` parameter must be provided in kilometers regardless of the unit of measure of the grid. The matrix `v_p` describes the cross-correlation between multiple variables and it is equal to 1 for the univariate case.

At this point, model estimation can be performed by calling the method `EM_estimate` of the class '`stem_model`' which requires as input argument an object of class '`stem_EM_options`'.

```
>> exit_toll = 0.001;
>> max_iterations = 100;
>> obj_stem_EM_options = stem_EM_options(exit_toll, max_iterations);
>> obj_stem_model.EM_estimate(obj_stem_EM_options);
>> obj_stem_model.set_varcov;
>> obj_stem_model.set_logL;
```

The variance-covariance matrix of the estimated model parameters and the observed data log-likelihood are evaluated after model estimation using the methods `set_varcov` and `set_logL` of the class `stem_model`. All the relevant information about model estimation can be found in the internal object `stem_EM_result` which can be accessed as a property of the `obj_stem_model` object. After model estimation, the `obj_stem_model` object is saved in the subfolder `Output` of the `Demo` folder.

Using the `print` method of class '`stem_model`', the following output is obtained.

```
*********************************
*   Model estimation results    *
*********************************
* Tapering is not enabled

* Observed data log-likelihood: -13889.064

* Beta coefficients related to the point variable no2 ground
    'Loading coefficient'    'Value'     'Std'
    'wind speed'             '-0.175'    '0.004'
    'elevation'              '-0.284'    '0.008'
    'sunday'                 '-0.102'    '0.007'

* Sigma_eps diagonal elements (Variance)
    'Variable'       'Value'    'Std'
    'no2 ground'     '0.392'    '0.002'
```

```
* 1 fine-scale coregionalization components w_p

* alpha_p elements:
              []    'elevation'
    'no2 ground'     '+0.590 (Std 0.005)'

* theta_p elements:
    'Coreg. component'    'Value [km]'    'Std [km]'
    '1st'                 '31.42'         '0.83'

* v_p matrix for the 1st coreg. component:
              []    'no2 ground'
    'no2 ground'     '+1.00'

* Transition matrix G:
                         []    'no2 ground - constant'
    'no2 ground - constant'     '+0.95 (Std 0.02)'


* Sigma_eta matrix:
                         []    'no2 ground - constant'
    'no2 ground - constant'     '+0.03 (Std 0.01)'
```

The standard deviations related to each estimated model parameter are directly obtained from the diagonal elements of the variance-covariance matrix.

The estimate of the latent temporal variable $z(t)$ is stored in the `stem_kalmansmoother_result` object which is a property of the `stem_EM_result` object. The graph of Figure 1 shows the estimated temporal variable and it has been obtained calling the method `plot` of class '`stem_kalmansmoother_result`'.

The estimated model is eventually used to map the NO_2 concentration over the geographic region following the kriging approach. Since the loading coefficients of this case study consist of a set of covariates, the same covariates must be available for the entire region as a regular grid with the proper spatial resolution.

The first step toward mapping is to create the `obj_stem_krig` object of class '`stem_krig`' in the following way.

```
>> obj_stem_krig = stem_krig(obj_stem_model);
```

Since the kriging output is evaluated over a regular grid, the `obj_stem_krig_grid` object of class '`stem_grid`' is created as follows.

```
>> load ../Data/kriging/krig_elevation_005;
>> krig_coordinates = [krig_elevation.lat(:), krig_elevation.lon(:)];
>> obj_stem_krig_grid = stem_grid(krig_coordinates, 'deg', 'regular', ...
     'pixel', [80, 170], 'square', 0.05, 0.05);
```

As the grid is regular, the dimension of the grid must also be provided (80 rows and 170 columns) as well as the shape of the pixels (square) and their dimension (0.05×0.05 degrees).

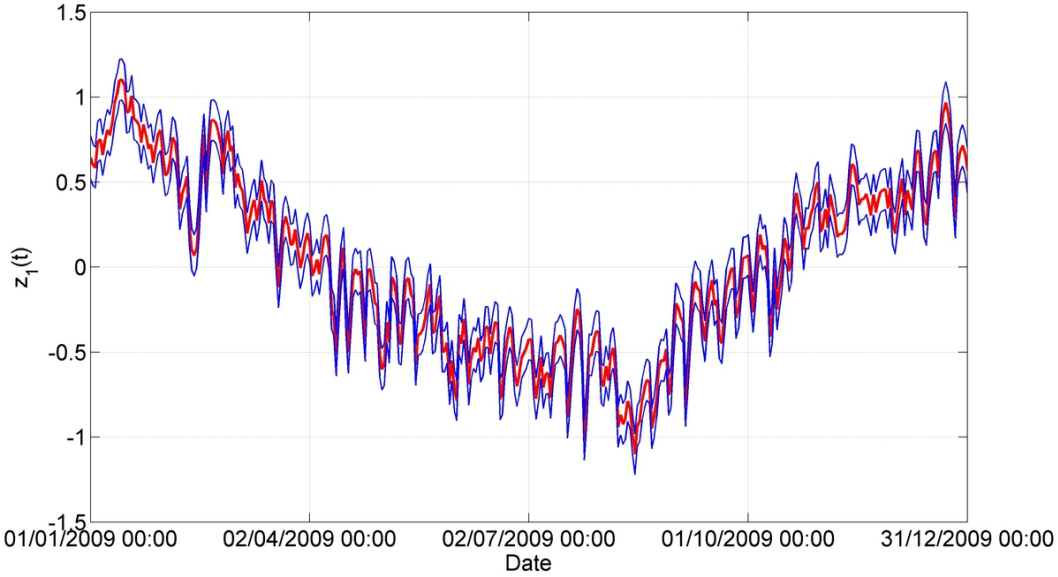

Figure 1: Estimated latent variable $z(t)$ and 95% confidence interval for the univariate model.

At this point, two important aspects must be considered. The first one is related to the grid pixels. If the variable should not be predicted over some areas of the geographic region, then it is possible to provide a mask of the pixels that must be excluded. This allows to reduce computing time.

The second aspect is related to the dimension of the grid and memory usage. If the grid is large and/or very dense, the number of pixels can be high and the loading coefficients may require a lot of memory when loaded. In order to avoid memory problems, the loading coefficients (related to the non-masked pixels) can be saved on disk within different blocks. Kriging is then executed block by block without the need of loading the entire data set of loading coefficients. On the other hand, when pixels are low in number, the user can implement kriging providing all the coefficients at once. All the details about the two approaches are found within the help of the class '`stem_krig`'. In this paper, the first approach is considered as more complex in terms of data structure.

Kriging is executed by calling the method `kriging` of the class '`stem_krig`' as it follows.

```
>> krig_mask = krig_elevation.data_mask(:);
>> back_transform = 1;
>> no_varcov = 0;
>> block_size = 1000;
>> X_krig = '../Data/kriging/blocks';
>> obj_stem_krig_result = obj_stem_krig.kriging('no2 ground', ...
     obj_stem_krig_grid, block_size, krig_mask, ...
     X_krig, back_transform, no_varcov);
```

If the observations have been log-transformed and/or standardized, the `back_transform` argument allows to produce the kriging output in the original unit of measure. If it is not necessary to estimate the variance of the prediction, the `no_varcov` argument can be set to

1 saving computing time. Finally, note that the directory where the blocks are stored is provided as well as the size (number of grid pixels) of each block.

The kriging result is saved in the `obj_stem_krig_result` object and the `plot` method can be used to display the result on a map. For example, the estimated NO_2 concentration and the respective standard deviation for April 10, 2009 are depicted in Figure 2. Note that, since $w(\mathbf{s}, t)$ and $x_{land}(\mathbf{s})$ are interacted, the spatial pattern of the standard deviation does not reflect the monitoring network, that is, the standard deviation is not necessarily lower near the monitoring stations.

5. Multivariate model

In order to demonstrate the multivariate capabilities of **D-STEM**, a simple bivariate case is introduced. A more complex case study with three variables can be found in Finazzi *et al.* (2013).

In addition to the NO_2 data of the previous section, measurements of particulate matters concentration $PM_{2.5}$ coming from $n_2 = 44$ monitoring stations over the same geographic region are considered. Note that only a subset of the $PM_{2.5}$ measurements are co-located to the NO_2 measurements. **D-STEM**, however, allows for fully or partially heterotopic networks. In this way, the spatial information of the more dense NO_2 monitoring network may be used to improve $PM_{2.5}$ mapping.

The response variable is now $\mathbf{y}(\mathbf{s}, t) = (y_{NO_2}(\mathbf{s}, t), y_{PM_{2.5}}(\mathbf{s}, t))^\top$ and the model in Equation 1 is extended by introducing a bivariate temporal component $\mathbf{z}(t)$ and a bivariate space-time component $\mathbf{w}(\mathbf{s}, t)$ modeled through an LCM. The same covariates of the previous section are considered for both NO_2 and $PM_{2.5}$.

5.1. Model description

Multivariate models are tackled considering the following straightforward extension of the model in Equation 2:

$$\mathbf{y}(\mathbf{s}, t) = \boldsymbol{\mu}(\mathbf{s}, t) + \boldsymbol{\omega}(\mathbf{s}, t) + \boldsymbol{\varepsilon}(\mathbf{s}, t), \tag{6}$$

which unifies the modeling approaches developed in Zhang (2007), Fassò and Finazzi (2011) and Finazzi *et al.* (2013).

In particular, extending fixed and random effect models of Equations 3 and 4, we have $\boldsymbol{\mu}(\mathbf{s}, t) = \mathbf{X}_{\boldsymbol{\beta}}(\mathbf{s}, t)\boldsymbol{\beta}$ and

$$\boldsymbol{\omega}(\mathbf{s}, t) = \sum_{j=1}^{c} \boldsymbol{\alpha}_j \odot \mathbf{x}_j(\mathbf{s}, t) \odot \mathbf{w}_j(\mathbf{s}, t) + \mathbf{X}_{\mathbf{z}}(\mathbf{s}, t)\mathbf{z}(t), \tag{7}$$

where the symbol $\odot$ represents the element by element or Hadamard product; moreover $\mathbf{y}(\mathbf{s}, t)$, $\boldsymbol{\alpha}_j$, $\mathbf{x}_j(\mathbf{s}, t)$, $\mathbf{w}_j(\mathbf{s}, t)$, $j = 1, \ldots, c$ and $\boldsymbol{\varepsilon}(\mathbf{s}, t)$ are $q \times 1$ vectors, while

$$\begin{aligned} \mathbf{X}_{\boldsymbol{\beta}}(\mathbf{s}, t) &= \text{blockdiag}(\mathbf{x}_{\boldsymbol{\beta},1}(\mathbf{s}, t), \ldots, \mathbf{x}_{\boldsymbol{\beta},q}(\mathbf{s}, t)), \\ \mathbf{X}_{\mathbf{z}}(\mathbf{s}, t) &= \text{blockdiag}(\mathbf{x}_{\mathbf{z},1}(\mathbf{s}, t), \ldots, \mathbf{x}_{\mathbf{z},p}(\mathbf{s}, t)), \end{aligned}$$

where blockdiag is the block diagonal building operator. The vectors of loading coefficients $\mathbf{x}_{\boldsymbol{\beta},i}(\mathbf{s}, t)$ have dimensions $1 \times b_i$ for $i = 1, \ldots, q$ while the vectors $\mathbf{x}_{\mathbf{z},k}(\mathbf{s}, t)$ have dimensions $1 \times a_k$ for $k = 1, \ldots, p$.

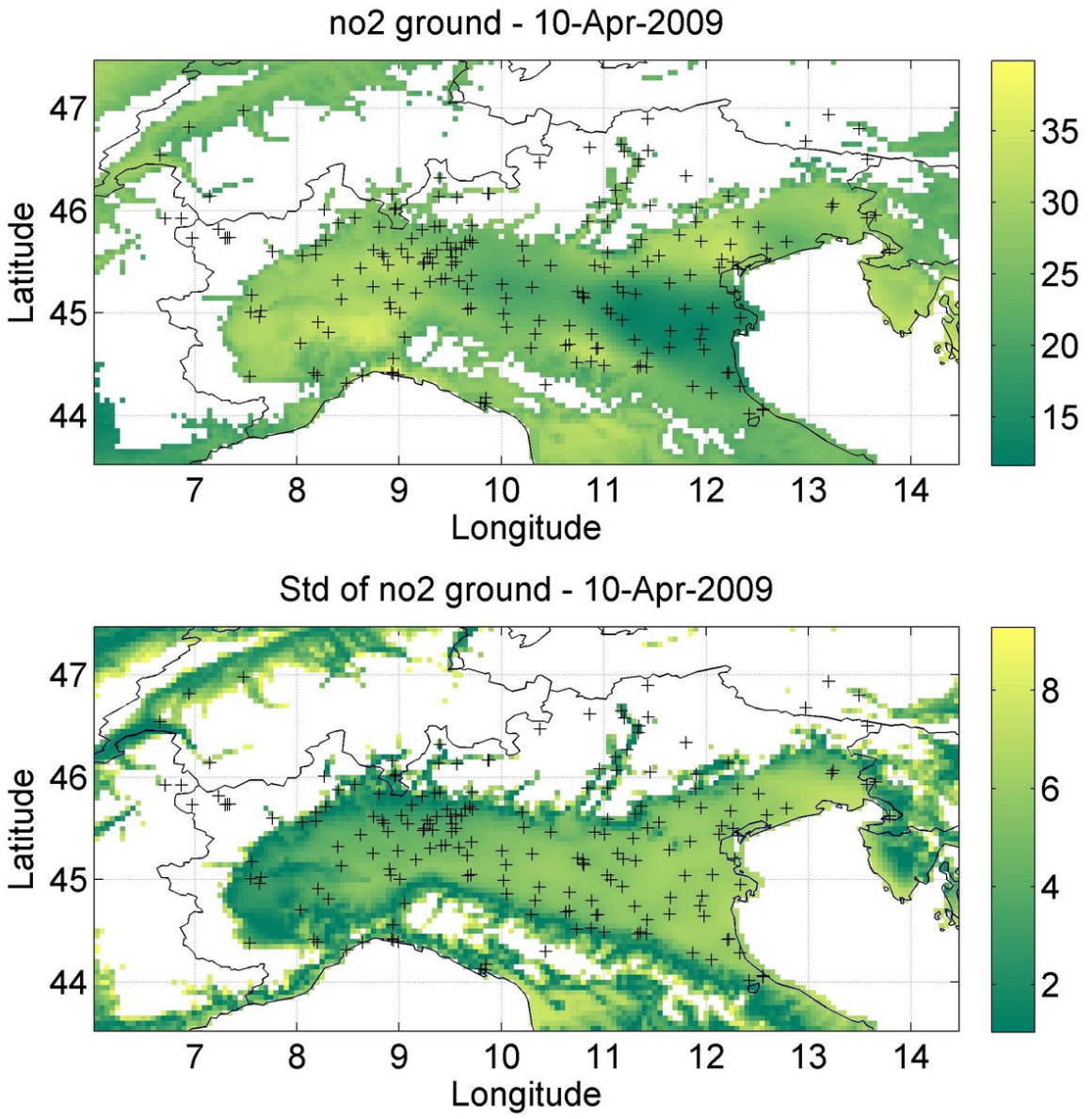

Figure 2: Estimated NO_2 concentration $[\mu g \cdot m^{-3}]$ below 800 m of elevation (top) and standard deviation (bottom) for April 10, 2009 over Northern Italy using the univariate model. Monitoring stations are depicted by the '+' symbol.

In Equation 7, each multivariate spatial latent variable $\mathbf{w}_j(\mathbf{s}, t)$, for each fixed t, is modeled as a LCM with the following spatial variance-covariance matrix functions

$$\mathbf{\Gamma}_j(\|\mathbf{s} - \mathbf{s}'\|) = \mathbf{V}_j \rho(\|\mathbf{s} - \mathbf{s}'\|; \theta_j, \nu), \tag{8}$$

where $\mathbf{V}_j$ is a valid $q \times q$ correlation matrix and $j = 1, \ldots, c$. On the other hand, the elements of $\boldsymbol{\varepsilon}(\mathbf{s}, t)$ are independent and normally distributed with variances σ_i^2, $i = 1, \ldots, q$. It follows

that the parameter set for the model in Equation 6 is

$$\psi = \{\boldsymbol{\beta}, \boldsymbol{\sigma}^2, \boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{v}, \mathbf{G}, \mathbf{v}_{\boldsymbol{\eta}}\},$$

where $\boldsymbol{\sigma}^2 = (\sigma_1^2, \ldots, \sigma_q^2)^\top$, $\boldsymbol{\alpha}$ is the $cq \times 1$ dimensional vector obtained by stacking $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_c$ and $\mathbf{v}$ is the $cq(q-1)/2 \times 1$ dimensional vector obtained by stacking the unique and non diagonal elements of $\mathbf{V}_1, \ldots, \mathbf{V}_c$.

5.2. Software implementation

This paragraph describes the relevant lines of code of the `demo_section5.m` script. The script can be executed choosing option number two from the `dstem_demo.m` script. Since `demo_section4.m` of Section 4 and `demo_section5.m` are similar, only the differences induced by the multivariate setting are detailed here.

The additional data related to the $PM_{2.5}$ variable are loaded into the temporary structure `ground` in the following way.

```
>> load ../Data/pm25_ground.mat
>> ground.Y{2} = pm25_ground.data;
>> ground.Y_name{2} = 'pm2.5 ground';
>> n2 = size(ground.Y{2}, 1);
```

Note that the second cell of the cell arrays `Y` and `Y_name` is used. The same strategy is followed for `X_beta`, `X_beta_name` and so further. The coordinates of each variable must be provided separately and added to the `obj_stem_gridlist_p` object as follows.

```
>> ground.coordinates{1} = [no2_ground.lat, no2_ground.lon];
>> ground.coordinates{2} = [pm25_ground.lat, pm25_ground.lon];
>> obj_stem_grid1 = stem_grid(ground.coordinates{1}, 'deg', 'sparse', ...
     'point');
>> obj_stem_grid2 = stem_grid(ground.coordinates{2}, 'deg', 'sparse', ...
     'point');
>> obj_stem_gridlist_p.add(obj_stem_grid1);
>> obj_stem_gridlist_p.add(obj_stem_grid2);
```

As in Section 4.2, the suffix "`_p`" refers to ground data because it is necessary to differentiate then from the remote data of Section 6. Now, the `obj_stem_par` object is created in the following way.

```
>> flag_time_diagonal = 0;
>> obj_stem_par = stem_par(obj_stem_data, 'exponential', [], ...
     flag_time_diagonal);
```

Here, $\mathbf{z}(t)$ is bivariate ($p = 2$) and the `flag_time_diagonal` flag is introduced and used to specify if the matrices $\mathbf{G}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$ are diagonal or not. The following lines of code describe the initialization of the model parameters for the bivariate case.

```
>> obj_stem_par.beta = obj_stem_model.get_beta0();
>> obj_stem_par.alpha_p = [0.6 0.6]';
```

```
>> obj_stem_par.theta_p = 100;
>> obj_stem_par.v_p = [1 0.6; 0.6 1];
>> obj_stem_par.sigma_eta = diag([0.2 0.2]);
>> obj_stem_par.G = diag([0.8 0.8]);
>> obj_stem_par.sigma_eps = diag([0.3 0.3]);
>> obj_stem_model.set_initial_values(obj_stem_par);
```

Model estimation is thus obtained as in the previous section and it gives the following results.

```
*******************************
*   Model estimation results   *
*******************************
* Tapering is not enabled

* Observed data log-likelihood: -14468.064

* Beta coefficients related to the point variable no2 ground
    'Loading coefficient'    'Value'     'Std'
    'wind speed'             '-0.174'    '0.004'
    'elevation'              '-0.285'    '0.008'
    'sunday'                 '-0.103'    '0.007'


* Beta coefficients related to the point variable pm2.5 ground
    'Loading coefficient'    'Value'     'Std'
    'wind speed'             '-0.150'    '0.007'
    'elevation'              '-0.203'    '0.009'
    'sunday'                 '-0.012'    '0.013'

* Sigma_eps diagonal elements (Variance)
    'Variable'        'Value'    'Std'
    'no2 ground'      '0.394'    '0.002'
    'pm2.5 ground'    '0.322'    '0.004'

* 1 fine-scale coregionalization components w_p

* alpha_p elements:
                []    'elevation'
    'no2 ground'       '+0.603 (Std 0.005)'
                []    'elevation'
    'pm2.5 ground'     '+0.341 (Std 0.009)'

* theta_p elements:
    'Coreg. component'    'Value [km]'    'Std [km]'
    '1st'                 '37.08'         '0.95'

* v_p matrix for the 1st coreg. component:
```

```
                []    'no2 ground'          'pm2.5 ground'
    'no2 ground'       '+1.00'               '+0.70 (Std 0.02)'
    'pm2.5 ground'     '+0.70 (Std 0.02)'    '+1.00'

* Transition matrix G:
                []    'no2 ground'          'pm2.5 ground'
    'no2 ground'       '+1.00 (Std 0.03)'    '-0.05 (Std 0.02)'
    'pm2.5 ground '    '+0.29 (Std 0.07)'    '+0.72 (Std 0.05)'


* Sigma_eta matrix:
                []    'no2 ground'          'pm2.5 ground'
    'no2 ground'       '+0.03 (Std 0.01)'    '+0.03 (Std 0.01)'
    'pm2.5 ground'     '+0.03 (Std 0.01)'    '+0.10 (Std 0.01)'
```

Figure 3 shows the components of the estimated $\mathbf{z}(t)$ related to both the variables. The graphs are obtained by calling the method `plot` of the class '`stem_kalmansmoother_result`'. Daily concentration maps of both variables can be obtained as in the previous section.

6. Data fusion model

In order to demonstrate the data fusion capability of **D-STEM**, the case study of the previous section is extended by introducing remote sensing observations covering the same geographic region. In particular, the tropospheric column density of NO_2 (see Fassò and Finazzi 2013) and the so called aerosol optical thickness (AOT), which is known to be related to the ground level concentration of $PM_{2.5}$ (Wang and Christopher 2003), are considered. The data are provided as daily block averages over a regular grid with spatial resolution $1/4°$ that covers the entire globe.

Remote sensing data may represent a valuable data source for estimating the ground level variables after statistical calibration based on the available ground level data. In doing this, a change of support problem (COSP) must be solved (Gotway and Young 2002). Depending on the aim of the data analysis, the COSP can also be considered as a data fusion or downscaling problem. Applications are not restricted to air quality remote sensing but include the case of physical model outputs and the case of environmental areal data in general.

The next paragraph discusses a general data fusion model suitable to jointly model ground level and remote sensing data solving the COSP.

6.1. Model description

The above remote sensing data can be considered a special case of block or pixel data which are denoted here by $\boldsymbol{y}^{\mathsf{B}}(B,t)$, where $B \subset \mathcal{D}$ is the generic grid pixel. With this notation, the model of Equation 6 is extended by introducing a new equation for remote sensing data and a downscaling link term into the ground data equation as follows:

$$\boldsymbol{y}^{\mathsf{B}}(B,t) = \boldsymbol{\mu}^{\mathsf{B}}(B,t) + \boldsymbol{\omega}^{\mathsf{B}}(B,t) + \boldsymbol{\varepsilon}^{\mathsf{B}}(B,t) \tag{9}$$

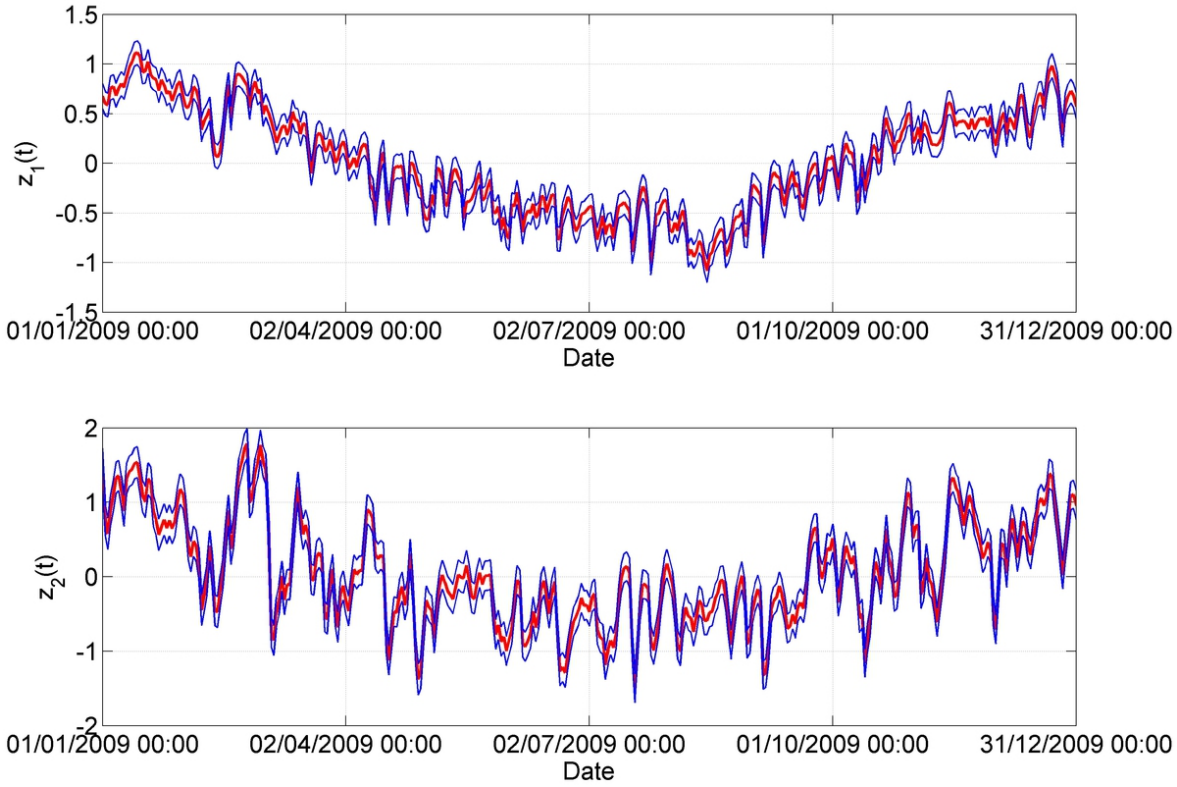

Figure 3: Estimated latent variable $\mathbf{z}(t)$ and 95% confidence interval: NO_2 component (top) and $PM_{2.5}$ component (bottom).

and

$$\boldsymbol{y}(\mathbf{s},t) = \boldsymbol{\mu}(\mathbf{s},t) + \boldsymbol{\omega}(\mathbf{s},t) + \tilde{\boldsymbol{\omega}}(\mathbf{s},t) + \boldsymbol{\varepsilon}(\mathbf{s},t). \tag{10}$$

In Equation 9, the fixed and random effects have a structure similar to Equation 6 with $c = 1$, namely

$$\boldsymbol{\mu}^{\mathsf{B}}(B,t) = \mathbf{X}^{\mathsf{B}}_{\boldsymbol{\beta}}(B,t)\boldsymbol{\beta}^{\mathsf{B}}$$

and

$$\boldsymbol{\omega}^{\mathsf{B}}(B,t) = \boldsymbol{\alpha}^{\mathsf{B}} \odot \mathbf{x}^{\mathsf{B}}(B,t) \odot \mathbf{w}^{\mathsf{B}}(B,t) + \mathbf{X}^{\mathsf{B}}_{\mathbf{z}}(B,t)\mathbf{z}^{\mathsf{B}}(t). \tag{11}$$

The resolution change between point and block data is defined by

$$\mathbf{w}^{\mathsf{B}}(B,t) = \frac{1}{|B|}\int_B \tilde{\mathbf{w}}(\mathbf{s},t)d\mathbf{s}, \tag{12}$$

where $\tilde{\mathbf{w}}(\mathbf{s},t)$ is a zero-mean Gaussian process with variance-covariance matrix function as defined in Equation 8 with parameters V^{B} and θ^{B}. Similarly to point data, the measurement error vector $\boldsymbol{\varepsilon}^{\mathsf{B}}(B,t)$ is assumed to be uncorrelated over space and time and across variables with variance vector $\boldsymbol{\sigma}^2_{\mathsf{B}} = (\sigma^2_{\mathsf{B},1},\ldots,\sigma^2_{\mathsf{B},q})^\top$.

In Equation 10, the additional remote-ground link term is given by

$$\tilde{\boldsymbol{\omega}}(\mathbf{s},t) = \boldsymbol{\alpha}^{\mathsf{BP}} \odot \mathbf{x}^{\mathsf{BP}}(\mathbf{s},t) \odot \tilde{\mathbf{w}}(\mathbf{s},t), \tag{13}$$

where, the parameter vector $\boldsymbol{\alpha}^{\mathsf{BP}}$ gives the intensity of the correlation between pixel and point variables, which can be modeled by the $q \times 1$ vector of loading coefficients $\mathbf{x}^{\mathsf{BP}}(\mathbf{s}, t)$.

Note that $\mathbf{z}^{\mathsf{B}}(t)$ is an additional Markovian component as the temporal dynamics may differ beween remote and ground data.

The parameter set for the data fusion model is

$$\boldsymbol{\psi} = \{\boldsymbol{\beta}, \boldsymbol{\beta}^{\mathsf{B}}, \boldsymbol{\sigma}^2, \boldsymbol{\sigma}^2_{\mathsf{B}}, \boldsymbol{\alpha}, \boldsymbol{\alpha}^{\mathsf{BP}}, \boldsymbol{\alpha}^{\mathsf{B}}, \theta, \mathbf{v}, \theta^{\mathsf{B}}, \mathbf{v}^{\mathsf{B}}, \mathbf{G}, \mathbf{v}_{\boldsymbol{\eta}}\},$$

where $\mathbf{v}^{\mathsf{B}}$ is the $q(q-1)/2 \times 1$ dimensional vector obtained by stacking the unique, non-diagonal elements of $\mathbf{V}^{\mathsf{B}}$.

In many environmental applications, the grid of pixels is regular (all the pixels have the same shape and dimension), the pixels are small compared to the geographic region and the process $\tilde{\mathbf{w}}(\mathbf{s}, t)$ is smooth. In this case the approximations $\mathbf{w}^{\mathsf{B}}(B, t) \simeq \tilde{\mathbf{w}}(\mathbf{s}^*, t)$ and $\mathbf{w}(\mathbf{s}, t) \simeq \tilde{\mathbf{w}}(\mathbf{s}^*, t)$, where $\mathbf{s}^*$ is the center of B, are reasonable. **D-STEM** implements these approximations avoiding the time-consuming computation of the integral in Equation 12 for each pixel B and each iteration of the EM algorithm. Possible errors induced by this approximation are covered by $\boldsymbol{\varepsilon}^{\mathsf{B}}(B, t)$.

The model in Equations 9 and 10 is similar to the Gaussian Markov random field smoothed downscaler developed in Berrocal, Gelfand, and Holland (2012), with the main difference that **D-STEM** handles multivariate data and use Gaussian processes instead of Gaussian Markov random fields. In fact Gaussian processes, thanks to the EM algorithm, are more suitable for handling extensive missing data which often arise in remote sensing.

6.2. Software implementation

This paragraph describes the `demo_section6.m` script which can be executed choosing option number three from the `dstem_demo.m` script. Only the code related to the pixel variables and the downscaler are discussed here, the reader being referred to the previous Section 5.2.

A simple version of the model in Equation 9 is considered here. In particular, remote sensing data are described by the equation

$$\mathbf{y}^{\mathsf{B}}(B, t) = \boldsymbol{\alpha}^{\mathsf{B}} \odot \mathbf{w}^{\mathsf{B}}(B, t) + \boldsymbol{\varepsilon}^{\mathsf{B}}(B, t), \tag{14}$$

and, similarly, $\mathbf{x}^{\mathsf{BP}}(\mathbf{s}, t) = \mathbf{1}$ is used in Equation 13. Hence a constant vector is added to the temporary data structure `ground` of Section 5.2 as follows.

```
>> ground.X_bp{1} = ones(n1, 1);
>> ground.X_bp_name{1} = {'constant'};
>> ground.X_bp{2} = ones(n2, 1);
>> ground.X_bp_name{2} = {'constant'};
```

The observations related to the pixel variables are loaded from disk as detailed in the following lines of code.

```
>> load ../Data/no2_remote_025.mat
>> remote.Y{1} = no2_remote.data;
>> remote.Y_name{1} = 'no2 remote';
```

```
>> m1 = size(remote.Y{1}, 1);
>> load ../Data/aot_remote_025.mat
>> remote.Y{2} = aot_remote.data;
>> remote.Y_name{2} = 'aot remote';
>> m2 = size(remote.Y{2}, 1);
```

Note that an additional temporary data structure `remote` is used. Since the model in Equation 14 does not consider loading coefficients for the pixel variables, the constant vector $\mathbf{x}^{\mathsf{B}}(B,t) = \mathbf{1}$ is provided in the following way.

```
>> remote.X_bp{1} = ones(m1, 1);
>> remote.X_bp_name{1} = {'constant'};
>> remote.X_bp{2} = ones(m2, 1);
>> remote.X_bp_name{2} = {'constant'};
```

A second object of class '`stem_varset`' is then created.

```
>> obj_stem_varset_b = stem_varset(remote.Y, remote.Y_name, remote.X_bp, ...
     remote.X_bp_name);
```

Following the same strategy, a second object of class '`stem_gridlist`' is created and it is used as a collector for the objects of class '`stem_grid`' related to the pixel variables.

```
>> obj_stem_gridlist_b = stem_gridlist();
>> remote.coordinates{1} = [no2_remote.lat(:), no2_remote.lon(:)];
>> remote.coordinates{2} = [aot_remote.lat(:), aot_remote.lon(:)];
>> obj_stem_grid1 = stem_grid(remote.coordinates{1}, 'deg', 'regular', ...
     'pixel', size(no2_remote.lat), 'square', 0.25, 0.25);
>> obj_stem_grid2 = stem_grid(remote.coordinates{2}, 'deg', 'regular', ...
     'pixel', size(aot_remote.lat), 'square', 0.25, 0.25);
>> obj_stem_gridlist_b.add(obj_stem_grid1);
>> obj_stem_gridlist_b.add(obj_stem_grid2);
```

When multiple pixel variables are considered, it is possible to decide whether $\mathbf{w}^{\mathsf{B}}(B,t)$ is cross-correlated or not. If not, then the spatial correlation function of each variable is parametrized by its own parameter vector θ_i^{B}, $i = 1, \ldots, q$. The additional `flag_pixel_correlated` flag is thus introduced and it is used as input argument in the creation of the `obj_stem_data` and `obj_stem_par` objects.

```
>> flag_pixel_correlated = 0;
>> flag_time_diagonal = 0;
>> obj_stem_data = stem_data(obj_stem_varset_p, obj_stem_gridlist_p, ...
     obj_stem_varset_b, obj_stem_gridlist_b, obj_stem_datestamp, ...
     [], [], [], flag_pixel_correlated);
>> obj_stem_par = stem_par(obj_stem_data, 'exponential', ...
     flag_time_diagonal);
>> obj_stem_model = stem_model(obj_stem_data,obj_stem_par);
```

The model parameters related to the pixel variables are initialized in the following way.

```
>> obj_stem_par.alpha_bp = [0.4 0.4 0.8 0.8]';
>> if flag_pixel_correlated
     obj_stem_par.theta_b = 100;
     obj_stem_par.v_b = [1 0.6; 0.6 1];
   else
     obj_stem_par.theta_b = [100 100]';
     obj_stem_par.v_b = eye(2);
   end
>> obj_stem_par.sigma_eps = diag([0.3 0.3 0.3 0.3]);
>> obj_stem_model.set_initial_values(obj_stem_par);
```

Note that, depending on the value of `flag_pixel_correlated`, the parameter structure is different. Moreover, `alpha_bp` is a $2q \times 1$ vector that includes both $\boldsymbol{\alpha}^{\mathsf{BP}}$ and $\boldsymbol{\alpha}^{\mathsf{B}}$ while `sigma_eps` is a $2q \times 2q$ diagonal matrix, where $2q$ is the total number of variables.

Model estimation and kriging are performed using the same lines of code detailed in the previous sections and the estimation result is the following.

```
*********************************
*   Model estimation results   *
*********************************
* Tapering is not enabled

* Observed data log-likelihood: 7572.506

* Beta coefficients related to the point variable no2 ground
    'Loading coefficient'    'Value'     'Std'
    'wind speed'             '-0.169'    '0.004'
    'elevation'              '-0.252'    '0.006'
    'sunday'                 '-0.092'    '0.007'

* Beta coefficients related to the point variable pm2.5 ground
    'Loading coefficient'    'Value'     'Std'
    'wind speed'             '-0.142'    '0.008'
    'elevation'              '-0.229'    '0.008'
    'sunday'                 '-0.018'    '0.014'

* Sigma_eps diagonal elements (Variance)
    'Variable'         'Value'    'Std'
    'no2 ground'       '0.383'    '0.002'
    'pm2.5 ground'     '0.272'    '0.004'
    'no2 remote'       '0.024'    '0.001'
    'aot remote'       '0.065'    '0.003'

* alpha_bp elements
    'Variable'         'Value'     'Std'
    'no2 ground'       '+0.121'    '0.004'
    'pm2.5 ground'     '+0.314'    '0.009'
```

```
    'no2 remote'      '+0.891'    '0.008'
    'aot remote'      '+0.999'    '0.015'

* Pixel data are NOT cross-correlated.

* Theta_b elements:
    'Variable'      'Value [km]'    'Std [km]'
    'no2 remote'    '118.896'       '2.610'
    'aot remote'    '156.490'       '5.348'

* 1 fine-scale coregionalization components w_p

* alpha_p elements:
                []    'elevation'
    'no2 ground'       '+0.555 (Std 0.005)'
                []    'elevation'
    'pm2.5 ground'     '+0.302 (Std 0.008)'

* theta_p elements:
    'Coreg. component'    'Value [km]'    'Std [km]'
    '1st'                 '21.97'         '0.62'

* v_p matrix for the 1st coreg. component:
                []    'no2 ground'          'pm2.5 ground'
    'no2 ground'       '+1.00'               '+0.71 (Std 0.03)'
    'pm2.5 ground'     '+0.71 (Std 0.03)'    '+1.00'

* Transition matrix G:
                []    'no2 ground'          'pm2.5 ground'
    'no2 ground'       '+0.95 (Std 0.04)'    '-0.01 (Std 0.03)'
    'pm2.5 ground'     '+0.37 (Std 0.08)'    '+0.63 (Std 0.06)'


* Sigma_eta matrix:
                []    'no2 ground'          'pm2.5 ground'
    'no2 ground'       '+0.03 (Std 0.01)'    '+0.03 (Std 0.01)'
    'pm2.5 ground'     '+0.03 (Std 0.01)'    '+0.10 (Std 0.01)'
```

The first q elements of the vector `alpha_bp` express how well the latent variable $\mathbf{w}^{\mathsf{B}}(B,t)$, which describes the pixel observations, is also able to describe the respective point observations (net of the other model terms). When observations and loading coefficients are standardized, a value close to zero implies poor correlation while a value close to one implies high correlation. In this case study the values are 0.121 and 0.314 for NO_2 and $PM_{2.5}$, respectively, which correspond to low/mild correlations.

Pixel variables are considered as secondary information useful to improve the mapping of the point variables. For this reason, kriging is not provided for pixel variables. Nevertheless, the estimated pixel variables, namely $\hat{\mathbf{w}}^{\mathsf{B}}(B,t)$, are obtained over the original grid as a by-

product of model estimation. The following lines of code are used to plot the observed pixel variable NO_2 and the estimated $\hat{\mathbf{w}}^{\mathsf{B}}(B,t)$ which is stored in the property `E_wb_y1` of the `stem_EM_result` object.

```
>> obj_stem_model.stem_data.plot('no2 remote', 'pixel', 25);
>> size = obj_stem_model.stem_data.stem_gridlist_b.grid{1}.grid_size;
>> E_wb_y1 = obj_stem_model.stem_EM_result.E_wb_y1(1:size(1) * size(2), 25);
>> coordinate = obj_stem_model.stem_data.stem_gridlist_b.grid{1}.coordinate;
>> lat = coordinate(:, 1);
>> lon = coordinate(:, 2);
>> lat = reshape(lat,size);
>> lon = reshape(lon,size);
>> E_wb_y1 = reshape(E_wb_y1, size);
>> stem_misc.plot_map(lat, lon, E_wb_y1, obj_stem_model.stem_data.shape, ...
     'no2 remote estimated on 25-Jan-2009', 'Longitude', 'Latitude');
```

The resulting maps are displayed in Figure 4. The observed pixel data are characterized by large areas of missing data but the latent variable $\mathbf{w}^{\mathsf{B}}(B,t)$ allows to reconstruct the missing data and to filter the observed data corrupted by noise. Moreover, since $\mathbf{w}^{\mathsf{B}}(B,t)$ is used to model both point data and pixel data, the reconstruction of the missing pixel data benefits from the observed point data.

7. Large data sets handling

The statistical models that **D-STEM** implements are separable with respect to space and time. If N is the total number of spatial locations where all the point variables are observed and T is the total number of time steps, then the largest variance-covariance matrix that **D-STEM** handles is only $N \times N$. If pixel variables are also considered and the total number of pixels where all the variables are observed is M, then the largest variance-covariance matrix is $D \times D$, where $D = \max(N, M)$. In many applications, however, N and D can be large and both computing time and memory usage can increase drastically. In the following paragraphs, three strategies that **D-STEM** provides for reducing computing time are discussed. Even if these strategies are intended for large data sets, in order to keep the computing time feasible, the same case studies of the previous sections are considered.

7.1. Tapering

The tapering approach consists of adopting a sparse variance-covariance matrix characterized by a high percentage of zero elements (possibly higher than 90%). The idea behind tapering is that spatial locations at great distance should not exhibit spatial correlation. Hence, tapering forces to zero the covariances of observations at distances higher than a threshold in such a way that the positive definiteness of the variance-covariance matrix is preserved.

When the generic variance-covariance matrix $\mathbf{A}$ is used to solve matrix equations in the form $\mathbf{Ax} = \mathbf{b}$, the computing time is greatly reduced if $\mathbf{A}$ is sparse. The higher the matrix sparseness the lower the computing time. In order to model spatial correlation, however, the above mentioned threshold cannot be too low and there exists a trade-off between low

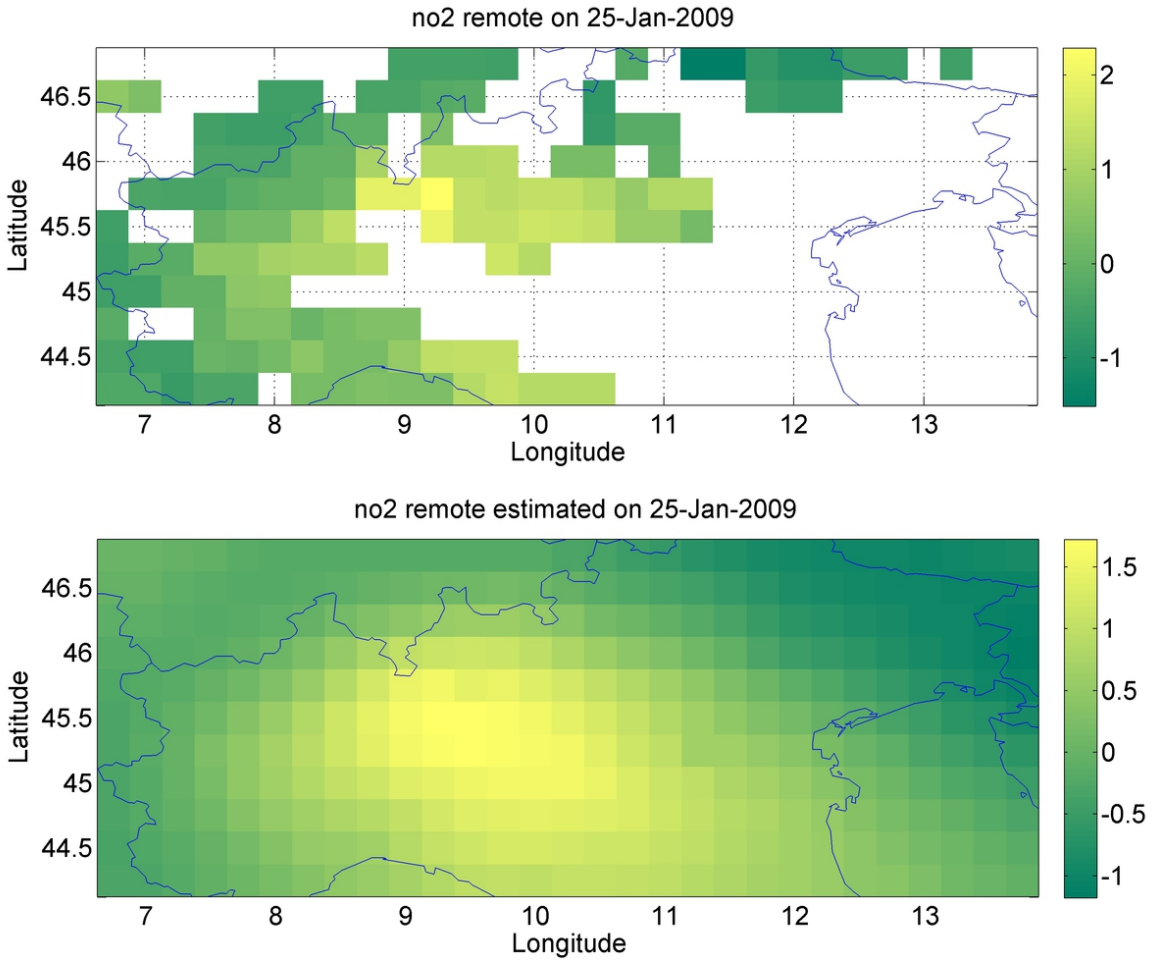

Figure 4: Observed remote sensing NO_2 data (top) and reconstructed data (bottom).

computing time and good approximation of the latent Gaussian processes used to describe the spatial correlation.

In order to implement tapering, **D-STEM** applies the so called one-taper tapering of Kaufman, Schervish, and Nychka (2008) to each LCM component $\mathbf{w}_j(\mathbf{s}, t)$. To do this, in the spatial correlation matrix function of Equation 8, the Matérn correlation function ρ is substituted by

$$\rho(\|\mathbf{s} - \mathbf{s}'\|; \theta_j, \nu) \cdot \Phi\left(\frac{\|\mathbf{s} - \mathbf{s}'\|}{\phi}\right),$$

where $\Phi\left(\frac{\|\mathbf{s}-\mathbf{s}'\|}{\phi}\right)$ is the compactly supported radial Wendland function (Wendland and Mathematik 1995), with ϕ the width of the radial function.

Although the asymptotic theory of Kaufman *et al.* (2008) is proven for the purely spatial univariate case, $T = 1$ and $q = 1$, in light of the results of Ruiz-Medina and Porcu (2014), we conjecture here that it holds true also for the purely spatial multivariate case, $T = 1$ and $q > 1$, and, a fortiori, for the space-time case with large T. Note that, unlike the so called two-tapers tapering of Kaufman *et al.* (2008), the one-taper may give a biased estimate of θ in

case the tapering range ϕ is small compared to the data correlation range. However, according to Kaufman *et al.* (2008), the two-tapers approach has a substantially heavier computational burden while the one-taper approach gives better kriging performance, which is an important aim of **D-STEM**.

The `demo_section7_1.m` script, which can be executed by choosing option number four from the `dstem_demo.m` script, implements the same case study of Section 6 with tapering enabled. In particular, tapering can be enabled by providing the width ϕ (expressed in kilometers) of the Wendland function to the constructor of the class '`stem_gridlist`' as follows.

```
>> phi_p = 50;
>> obj_stem_gridlist_p = stem_gridlist(phi_p);
>> phi_b = 200;
>> obj_stem_gridlist_b = stem_gridlist(phi_b);
```

The width ϕ is a property of the grids as all the variance-covariance matrices are directly derived from the distance matrices which, in order to reduce memory usage, are also created as sparse matrices. Also note that ϕ can be different for point and pixel data.

The output of model estimation is similar to the output reported in Section 6 and only the relevant part is reported hereafter.

```
*********************************
*   Model estimation results    *
*********************************
* Tapering is enabled.
  Point data tapering: 50 km
  Pixel data tapering: 200 km

* Observed data log-likelihood: 6182.198

* Theta_b elements:
    'Variable'      'Value [km]'    'Std [km]'
    'no2 remote'    '242.369'       ' 8.250'
    'aot remote'    '365.052'       '34.033'

* 1 fine-scale coregionalization components w_p

* theta_p elements:
    'Coreg. component'    'Value [km]'    'Std [km]'
    '1st'                 '198.22'        '80.21'
```

Looking at the estimation result, it can be noted that the estimated θ parameters, as well as their standard deviations, are higher compared to those reported in Section 6. Moreover, the observed data log-likelihood is lower. The tapering approach, thus, reduces computing time but may produce biased estimates of the θ parameters and/or estimates with a larger uncertainty. Finally, note that tapering is intended for large data sets. The case studies discussed in this paper are based on medium-size data sets so that the actual computing time may be higher when tapering is enabled. The same consideration applies to the two following strategies.

7.2. Computing load distribution

Thanks to the separability between space and time, most of the matrix algebra operations at the basis of the EM algorithm only involve the data of a single time step. Moreover, each time step is independent from the previous and the next time steps. The Kalman filter itself is implemented in such a way that some operations executed at time t are independent from the result of the operations at time $t-1$.

If a cluster of computing nodes is available, then model estimation can be performed in a distributed manner. In particular, if n nodes are available and $T \geq n$, then the data set is split into n temporal frames and distributed to the nodes on the basis of the node speed (evaluated at each EM iteration) and the number of missing data in each time frame. Indeed, time steps characterized by a high missing data rate imply faster computing.

The computing nodes can be any number of heterogeneous machines connected through a local area network (LAN) and no additional parallel and/or distributed software libraries are required. All the nodes must be able to read and write to a common shared folder and each node must run at least one MATLAB process. One node, usually the fastest or the one with the highest quantity of RAM, is designated to be the master while all the other nodes are considered slaves. The number of slaves can change during model estimation but the master node must always run.

In order to estimate the model in a distributed manner, a script that calls the `daemon.m` function must first be executed on each slave node possibly in batch mode. The function requires as input argument the path of the shared folder to use. The master runs the usual main script but the name of the shared folder, as well as additional parameters, must be given as properties of the `obj_stem_EM_options` object.

Choosing option number five from the `dstem_demo.m` script, the case study of Section 6 is reproduced and model estimation is carried out in distributed manner. To avoid the complication of setting up a distributed environment, the `dstem_demo.m` script starts a second MATLAB process in which the `demo_runslave` script is executed and that, in turn, executes the `daemon.m` script. The original MATLAB process executes the `demo_section7_2.m` script which differs from the `demo_section6.m` script with respect to the following lines of code.

```
>> exit_toll = 0.001;
>> max_iterations = 100;
>> path_distributed_computing = '../Distributed/';
>> timeout = 5;
>> obj_stem_EM_options = stem_EM_options(exit_toll, max_iterations, [], [], ...
     [], [], path_distributed_computing, [], timeout);
>> obj_stem_model.EM_estimate(obj_stem_EM_options);
```

The `timeout` input argument is the time in seconds that the master waits when listening for the slave nodes. It is worth knowing that the content of NFS shared folders on UNIX distributed environments is not always updated in real time. If the user cannot change the updating time of NFS folders, then the `timeout` input argument must be increased in order to ensure that master and slaves can always read the files written in the shared folder.

The output of model estimation is equal to the output already reported and discussed in Section 6.

7.3. Observed data log-likelihood evaluation

By default, **D-STEM** compares the estimated parameters and the observed data log-likelihood between two consecutive EM iterations. If the relative norm of the difference between the parameter vectors or between the log-likelihoods is lower than the tolerance specified by the property `exit_toll` of class '`stem_EM_options`' (see Section 4.2), the EM algorithm stops. In the case of large data sets, computing the log-likelihood at each EM iteration is time consuming. In order to speed up model estimation, the evaluation of the log-likelihood can be avoided and the exit condition is only based on the model parameters.

The `demo_section7_3.m` script, which can be executed choosing option number six from the `dstem_demo.m` script, implements the same case study as in Section 4 but model estimation is carried out without computing the observed data log-likelihood at each iteration.

The following lines of code describe how the `obj_stem_EM_options` object is created in order to avoid the evaluation of the log-likelihood at each iteration.

```
>> exit_toll = 0.001;
>> max_iterations = 100;
>> compute_log = 0;
>> obj_stem_EM_options = stem_EM_options(exit_toll, max_iterations, ...
     [], [], compute_log);
>> obj_stem_model.EM_estimate(obj_stem_EM_options);
```

The output of the model estimation is similar to the output reported in Section 4 and only the relevant part is reported hereafter.

```
*******************************
*   Model estimation results   *
*******************************
* Tapering is not enabled

* Observed data log-likelihood: -13797.357

* 1 fine-scale coregionalization components w_p

* theta_p elements:
    'Coreg. component'    'Value [km]'    'Std [km]'
    '1st'                 '22.45'         '0.63'
```

Due to the different exit condition, the model parameters estimated without computing the log-likelihood at each iteration differ from those estimated in Section 4. In particular, the θ parameter decreased from 31.42 to 22.45 km. Nonetheless, the observed data log-likelihood, evaluated after model estimation, is only slightly higher (-13797.357 vs. -13889.064). This is due to the fact that, using a non-large data set as in this case study, the θ parameter of Equation 5 is poorly identifiable and it monotonically changes from one iteration to the next even if the observed data log-likelihood does not change significantly.

Although the computing time of each iteration is reduced, thus, a possible drawback is that the total number of iterations required to estimate the model might be higher. The user

should be careful when adopting this strategy as, if poor identifiability of θ is not detected, the EM algorithm might not converge or it might take more iterations than necessary. In the above example, the EM algorithm takes 78 iterations to converge with respect to the 32 iterations required when the log-likelihood is evaluated at each iteration. Again, this strategy to reduce computing time is intended for large data sets.

8. Conclusions

In this paper, the use of **D-STEM** has been illustrated for three different case studies involving a univariate model, a bivariate model and a data fusion model. The model at the basis of **D-STEM** is general enough to accommodate many environmental data sets, nonetheless, both model and software can be extended with respect to many aspects. From the modeling point of view, additional spatial correlation functions could be introduced as well as more flexible "coregionalization models" (Apanasovich and Genton 2010; Gneiting, Kleiber, and Schlather 2010). Moreover, Markov random fields could be introduced in order to model pixel data. Indeed, Gaussian random fields easily handle data sets with extensive missing data but they are more computationally expensive even under tapering. Finally, it could be useful to extend the model to accommodate for time-varying grids and irregularly spaced sampling times.

Regardint the software side, some time consuming procedures such as the estimation of the variance-covariance matrix of the model parameters and kriging could also be implemented in a distributed manner. Furthermore, the handling of the model parameters could be improved by introducing constraints on the parameter vectors and matrices, widening the range of models that can be estimated.

D-STEM is constantly updated and improved and new versions are released on `https://code.google.com/p/d-stem/`. Google Code runs a project hosting service that provides revision control and an issue tracker. The users of **D-STEM** are welcome to notify bugs and to submit extensions or improvements of the code.

References

Apanasovich TV, Genton MG (2010). "Cross-Covariance Functions for Multivariate Random Fields Based on Latent Dimensions." *Biometrika*, **97**(1), 15–30.

Bakar KS, Sahu SK (2014a). "**spTimer**: Spatio-Temporal Bayesian Modeling Using R." *Journal of Statistical Software*. Forthcoming.

Bakar KS, Sahu SK (2014b). ***spTimer**: Spatio-Temporal Bayesian Modelling Using R*. R package version 1.0-3, URL `http://CRAN.R-project.org/package=spTimer`.

Berrocal VJ, Gelfand AE, Holland DM (2012). "Space-Time Data Fusion Under Error in Computer Model Output: An Application to Modeling Air Quality." *Biometrics*, **68**(3), 837–848.

Cameletti M (2012). ***Stem**: Spatio-Temporal Models in R*. R package version 1.0, URL `http://CRAN.R-project.org/package=Stem`.

Cameletti M, Lindgren F, Simpson D, Rue H (2013). "Spatio-Temporal Modeling of Particulate Matter Concentration through the SPDE Approach." *AStA Advances in Statistical Analysis*, **97**(2), 109–131.

Cressie NAC, Wikle CK (2011). *Statistics for Spatio-Temporal Data*. Wiley Series in Probability and Statistics. John Wiley & Sons.

Fassò A (2013). "Statistical Assessment of Air Quality Interventions." *Stochastic Environmental Research and Risk Assessment*, **27**(7), 1651–1660.

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

Fassò A, Finazzi F (2013). "A Varying Coefficients Space-Time Model for Ground and Satellite Air Quality Data over Europe." *Statistica & Applicazioni*, **Special Online Issue**, 45–56.

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.

Gneiting T, Kleiber W, Schlather M (2010). "Matérn Cross-Covariance Functions for Multivariate Random Fields." *Journal of the American Statistical Association*, **105**(491), 1167–1177.

Gotway CA, Young LJ (2002). "Combining Incompatible Spatial Data." *Journal of the American Statistical Association*, **97**(458), 632–648.

Katzfuss M, Cressie N (2011). "Spatio-Temporal Smoothing and EM Estimation for Massive Remote-Sensing Data Sets." *Journal of Time Series Analysis*, **32**(4), 430–446.

Kaufman CG, Schervish MJ, Nychka DW (2008). "Covariance Tapering for Likelihood-Based Estimation in Large Spatial Data Sets." *Journal of the American Statistical Association*, **103**(484), 1545–1555.

Pebesma E, Gaeler B (2013). ***gstat**: Spatial and Spatio-Temporal Geostatistical Modelling, Prediction and Simulation*. R package version 1.0-17, URL http://CRAN.R-project.org/package=gstat.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Rue H, Martino S, Lindgren F, 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/.

Ruiz-Medina MD, Porcu E (2014). "Equivalence of Gaussian Measures of Multivariate Random Fields." *Stochastic Environmental Research and Risk Assessment*. doi:10.1007/s00477-014-0926-z. Forthcoming.

The MathWorks, Inc (2010). *MATLAB, Statistics Toolbox, Optimization Toolbox and Mapping Toolbox Release 2010b*. Natick, Massachusetts, United States. URL http://www.mathworks.com/.

Wang J, Christopher SA (2003). "Intercomparison Between Satellite-Derived Aerosol Optical Thickness and PM2.5 Mass: Implications for Air Quality Studies." *Geophysical Research Letters*, **30**(21), 2095.

Wendland H, Mathematik A (1995). "Piecewise Polynomial, Positive Definite and Compactly Supported Radial Functions of Minimal Degree." *Advances in Computational Mathematics*, **4**(1), 389–396.

Zhang H (2007). "Maximum-Likelihood Estimation for Multivariate Spatial Linear Coregionalization Models." *Environmetrics*, **18**(2), 125–139.

Affiliation:

Francesco Finazzi
Department of Management, Economics and Quantitative Methods
University of Bergamo
via dei Caniana, 2
24127 Bergamo (BG), Italy
E-mail: francesco.finazzi@unibg.it
URL: http://www.unibg.it/pers/?francesco.finazzi

Alessandro Fassò
Department of Engineering
University of Bergamo
viale Marconi, 5
24044 Dalmine (BG), Italy
E-mail: alessandro.fasso@unibg.it
URL: http://www.unibg.it/pers/?alessandro.fasso

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 62, Issue 6 *Submitted:* 2013-08-05
December 2014 *Accepted:* 2014-07-11