A comparison of eight metamodeling techniques for the simulation of N2O fluxes and N leaching from corn crops

Nathalie Villa-Vialaneix, Marco Follador, Marco Ratto, Adrian Leip

To cite this version:

HAL Id: hal-00654753
https://hal.archives-ouvertes.fr/hal-00654753
Submitted on 22 Dec 2011
A comparison of eight metamodeling techniques for the simulation of $\text{N}_2\text{O}$ fluxes and $\text{N}$ leaching from corn crops

Nathalie Villa-Vialaneix$^{a,b,*}$, Marco Follador$^c$, Marco Ratto$^d$, Adrian Leip$^c$

$^a$IUT de Perpignan (Dpt STID, Carcassonne), Univ. Perpignan Via Domitia, France
$^b$Institut de Mathématiques de Toulouse, Université de Toulouse, France
$^c$European Commission, Institute for Environment and Sustainability, CCU, Ispra, Italy
$^d$European Commission, Econometrics and Applied Statistics Unit, Ispra, Italy

Abstract

The environmental costs of intensive farming activities are often underestimated or not traded by the market, even though they play an important role in addressing future society’s needs. The estimation of nitrogen (N) dynamics is thus an important issue which demands detailed simulation based methods and their integrated use to correctly represent complex and non-linear interactions into cropping systems. To calculate the $\text{N}_2\text{O}$ flux and $\text{N}$ leaching from European arable lands, a modeling framework has been developed by linking the CAPRI agro-economic dataset with the DNDC-EUROPE bio-geo-chemical model. But, despite the great power of modern calculators, their use at continental scale is often too computationally costly. By comparing several statistical methods this paper aims to design a metamodel able to approximate the expensive code of the detailed modeling approach, devising the best compromise between estimation performance and simulation speed. We describe the use of two parametric (linear) models and six nonparametric approaches: two methods based on splines (ACOSSO and SDR), one method based on kriging (DACE), a neural networks method (multilayer perceptron, MLP), SVM and a bagging method (random forest, RF). This analysis shows that, as long as few data are available to train the model, splines approaches lead to best results, while when the size of training dataset increases, SVM and RF provide faster and more accurate solutions.

*Corresponding author.

Email address: nathalie.villa@math.univ-toulouse.fr (Nathalie Villa-Vialaneix)
1. Introduction

The impact of modern agriculture on the environment is well documented (Power, 2010; Tilman et al., 2002; Scherr and Sthapit, 2009; FAO, 2007, 2005; Singh, 2000; Matson et al., 1997). Intensive farming has a high consumption of nitrogen, which is often in-efficiently used, particularly in livestock production systems (Leip et al., 2011b; Webb et al., 2005; Oenema et al., 2007; Chadwick, 2005). This leads to a large surplus of nitrogen which is lost to the environment. Up to 95% of ammonia emission in Europe have their origin in agricultural activities (Kirchmann et al., 1998; Leip et al., 2011a) contributing to eutrophication, loss of biodiversity and health problems. Beside NH$_3$, nitrate leaching below the soil root zone and entering the groundwater poses a particular problem for the quality of drinking water (van Grinsven et al., 2006). Additionally, agricultural sector is the major source of anthropogenic emissions of N$_2$O from the soils, mainly as a consequence of the application of mineral fertilizer or manure nitrogen (Del Grosso et al., 2006; Leip et al., 2011c; European Environment Agency, 2010; Leip et al., 2005). N$_2$O is a potent greenhouse gas (GHG) contributing with each kilogram emitted about 300 times more to global warming than the same mass emitted as CO$_2$, on the basis of a 100-years time horizon (Intergovernmental Panel on Climate Change, 2007).

Various European legislations attempt to reduce the environmental impact of the agriculture sector, particularly the Nitrates Directive (European Council, 1991) and the Water Framework Directive (European Council, 2000). Initially, however, compliance to these directives was poor (Oenema et al., 2009; European Commission, 2002). Therefore, with the last reform of the Common Agricultural Policy (CAP) in the year 2003 (European Council, 2003), the European Union introduced a compulsory Cross-Compliance (CC) mechanism to improve compliance with 18 environmental, food safety, animal welfare, and animal and plant health standards (Statutory Management Requirements, SMRs) as well as with requirements to maintain farmlands in good agricultural and environmental condition (Good Agricultural and Environment Condition requirements, GAECs), as prerequisite for receiving direct payments (European Union Commission, 2004; European Council,
The SMRs are based on pre-existing EU Directives and Regulations such as Nitrate Directives. The GAECs focus on soil erosion, soil organic matter, soil structure and a minimum level of maintenance; for each of these issues a number of standards are listed (Alliance Environnement, 2007).

It remains nevertheless a challenge to monitor compliance and to assess the impact of the cross-compliance legislations not only on the environment, but also on animal welfare, farmer’s income, production levels etc. In order to help with this task, the EU-project Cross-Compliance Assessment Tool (CCAT) developed a simulation platform to provide scientifically sound and regionally differentiated responses to various farming scenarios (Elbersen et al., 2010; Jongeneel et al., 2007).

CCAT integrates complementary models to assess changes in organic carbon and nitrogen fluxes from soils (De Vries et al., 2008). Carbon and nitrogen turnover are very complex processes, characterized by a high spatial variability and a strong dependence on environmental factors such as meteorological conditions and soils (Shaffer and Ma, 2001; Zhang et al., 2002). Quantification of fluxes, and specifically a meaningful quantification of the response to mitigation measures at the regional level requires the simulation of farm management and the soil/plant/atmosphere continuum at the highest possible resolution (Anderson et al., 2003; Leip et al., 2011c). For the simulation of N\textsubscript{2}O fluxes and N-leaching, the process-based biogeochemistry model DNDC-EUROPE (Leip et al., 2008; Li et al., 1992; Li, 2000) was used. As DNDC-EUROPE is a complex model imposing high computational costs, the time needed to obtain simulation results in large scale applications (such as the European scale) can be restrictive. In particular, the direct use of the deterministic model is prohibited to extract efficiently estimations of the evolution of N\textsubscript{2}O fluxes and N-leaching under changing conditions. Hence, there is a need for a second level of abstraction, modeling the DNDC-EUROPE model itself, which is called a meta-model (see Section 2 for a more specific definition of the concept of metamodeling). Metamodels are defined from a limited number of deterministic simulations for specific applications and/or scenario and allow to obtain fast estimations.

This issue is a topic of high interest that has previously been tackled in several papers: among others, (Bouzaher et al., 1993) develop a parametric model, including spatial dependency, to model water pollution. (Krysanova and Haberlandt, 2002; Haberlandt et al., 2002) describe a two-steps approach
to address the issue of N leaching and water pollution: they use a process-based model followed by a location of the results with a fuzzy rule. More recently, (Pineros Garcet et al., 2006) compare RBF neural networks with kriging modeling to build a metamodel for a deterministic N leaching model called WAVE (Vanclooster et al., 1996). The present article compares in detail different modeling tools in order to select the most reliable one to meta-model the DNDC-EUROPE tasks in the CCAT project Follador and Leip (2009). This study differs from the work of Vanclooster et al. (1996) because of the adopted European scale and of the analysis of 8 meta-modeling approaches (also including a kriging and a neural network method). The comparison has been based on the evaluation of meta-model performances, in terms of accuracy and computational costs, with different sizes of the training dataset.

The rest of the paper is organized as follows: Section 2 introduces the general principles and advantages of using a meta-model; Section 3 reviews in details the different types of metamodels compared in this study; Section 4 explains the Design Of the Experiments (DOE) and show the results of the comparison, highlighting how the availability of the training data can play an important role in the selection of the best type and form of the approximation. The supplementary material of this paper can be found at: http://afoludata.jrc.ec.europa.eu/index.php/dataset/detail/232.

2. From model to metamodel

A model is a simplified representation (abstraction) of reality developed for a specific goal; it may be deterministic or probabilistic. An integrated use of simulation based models is necessary to approximate our perception of complex and nonlinear interactions existing in human-natural systems by means of mathematical input-output (I/O) relationships. Despite the continuous increase of computer performance, the development of large simulation platforms remains often prohibited because of computational needs and parametrization constraints. More precisely, every model in a simulation platform such as DNDC-EUROPE, is characterized by several parameters, whose near-optimum set is defined during the calibration. A constraint applies restrictions to the kind of data that the model can use or to specific boundary conditions. The flux of I/O in the simulation platform can thus be impeded by the type of data/boundaries that constraints allow - or not allow - for the models at hand.
The use of this kind of simulation platform is therefore not recommended for all the applications which require many runs, such as sensitivity analysis or what-if studies. To overcome this limit, the process of abstraction can be applied to the model itself, obtaining a model of the model (2nd level of abstraction from reality) called meta-model (Blanning, 1975; Kleijnen, 1975; Sacks et al., 1989; van Gighc, 1991; Santner et al., 2003). A metamodel is an approximation of detailed model I/O transformations, built through a moderate number of computer experiments.

Replacing a detailed model with a metamodel generally brings some payoffs (Britz and Leip, 2009; Simpson et al., 2001):

- easier integration into other processes and simulation platforms;
- faster execution and reduced storage needs to estimate one specific output;
- easier applicability across different spatial and/or temporal scales and site-specific calibrations, as long as data corresponding to the new system parametrization are available.

As a consequence, a higher number of simulation runs become possible: using its interpolatory action makes a thorough sensitivity analysis more convenient and leads to a better understanding of I/O relationships. Also it offers usually a higher flexibility and can quickly be adapted to achieve a wide range of goals (prediction, optimization, exploration, validation). However, despite these advantages, they suffer from a few drawbacks: internal variables or outputs not originally considered can not be inspected and the prediction for input regimes outside the training/test set is impossible. Hence, a good metamodeling methodology should be able to provide fast predictions. But, considering these limitations, it also must have a low computational cost to be able to build a new metamodel from a new data set including new variables and/or a different range for these input variables.

Let \((X, y)\) be the dataset consisting of \(N\) row vectors of input/output pairs \((x_i, y_i)\), where \(x_i = (x_1^i, \ldots, x_d^i)^T \in \mathbb{R}^d\) \((i = 1, \ldots, N)\) are the model input and \(y_i \in \mathbb{R}\) \((i = 1, \ldots, N)\) are the model responses for \(N\) experimental runs of the simulation platform. The mathematical representation of I/O relationships described by the detailed model can be written as

\[
y_i = f(x_i) \quad i = 1, \ldots, N
\]
which corresponds to a first abstraction from the real system. From the
values of \( \mathbf{X} \) and \( \mathbf{y} \), also called training set, \( f \) is approximated by a function \( \hat{f} : \mathbb{R}^d \to \mathbb{R} \), called metamodel, whose responses can be written as
\[
\hat{y}_i = \hat{f}(\mathbf{x}_i).
\]
and that correspond to a second abstraction from the reality. In this second
abstraction, some of the input variables of Eq. (1) might not be useful and
one of the issue of metamodeling can be to find the smallest subset of input
variables relevant to achieve a good approximation of model (1).

Finally, the differences between the real system and the metamodel re-
sponse, will be the sum of two approximations (Simpson et al., 2001): the
first one introduced by the detailed model (1st abstraction) and the second
one due to metamodeling (2nd abstraction). Of course, the validity and ac-
curacy of a metamodel are conditioned by the validity of the original model:
in the following, it is then supposed that the 1st level of abstraction induces
a small error compared to reality. Then, in this paper, we only focus on
the second error, \( |\hat{y}_i - y_i| \), to assess the performance of different metamodels
vs. the detailed DNDC-EUROPE model in order to select the best statisti-
cal approach to approximate the complex bio-geo-chemical model at a lower
computational cost. Defining a correct metamodeling strategy is very impor-
tant to provide an adequate fitting to the model, as suggested by (Kleijnen
and Sargent, 2000; Meckesheimer et al., 2002).

Recent work, such as (Forrester and Keane, 2009; Wang and Shan, 2007),
review the most widely used metamodeling methods: splines based methods
(e.g., MARS, kriging...) (Wahba, 1990; Friedman, 1991; Cressie, 1990), neu-
ral networks (Bishop, 1995), kernel methods (SVM, SVR...) (Vapnik, 1998;
Christmann and Steinwart, 2007), Gaussian Process such as GEM (Kennedy
and O’Hagan, 2001), among others. Some of these metamodeling strategies
were selected and others added to be compared in this paper. The compar-
ison is made on a specific case study related to N leaching and N\textsubscript{2}O fluxes
prediction which is described in Section 4. The next section briefly describes
each of the metamodels compared in this paper.

3. Review of the selected metamodels

Several methods were developed and compared to assess their perfor-
mance according to increasing dataset sizes. We provide a brief description
of the approaches studied in this paper: two linear models (Section 3.1) and six nonparametric methods (two based on splines, in Sections 3.2.1 and 3.2.2, one based on a kriging approach, in Section 3.2.3, which is known to be efficient when analyzing computer experiments, a neural network method, in Section 3.2.4, SVM, in Section 3.2.5 and random forest, in Section 3.2.6).

3.1. Linear methods

The easiest way to handle the estimation of the model given in Eq. (1) is to suppose that $f$ has a simple parametric form. For example, the linear model supposes that $f(x) = \beta^T x + \beta_0$ where $\beta \in \mathbb{R}^d$ is a vector and $\beta_0$ is a real number, both of them have to be estimated from the observations $((x_i, y_i))_i$. An estimate is given by minimizing the sum of the square errors

$$\sum_{i=1}^N (y_i - (\beta^T x_i + \beta_0))^2$$

which leads to $\hat{\beta} = (X^T X)^{-1} X^T y$ and $\hat{\beta}_0 = \bar{y} - \hat{\beta}^T \bar{X}$ with $\bar{y} = \frac{1}{N} \sum_{i=1}^N y_i$ and $\bar{X} = \frac{1}{N} \sum_{i=1}^N x_i$.

In this paper two linear models were used:

- in the first one, the explanatory variables were the 11 inputs described in Section 4.2. This model is referred as “LM1”;

- the second one has been developed starting from the work of (Britz and Leip, 2009), that includes the 11 inputs of Section 4.2 but also their non linear transformations (square, square root, logarithm) and interaction components. A total of 120 coefficients were involved in this approach which is denoted by “LM2”. Including transformations and combinations of the 11 inputs has been designed in an attempt to better model a possible nonlinear phenomenon of the original model.

In the second case, due to the large number of explanatory variables, the model can be over-specified, especially if the training set is small. Actually, if the dimensionality of the matrix of explanatory variables, $X$, has a large dimension, $X^T X$ can be not invertible or ill-conditioned (leading to numerical instability). Hence, a stepwise selection based on the AIC criterion (Akaike, 1974) has been used to select an optimal subset of explanatory variables during the training step in order to obtain an accurate solution having a small number of parameters. This has been performed by using the `stepAIC` function of the R package `MASS`. 
3.2. Nonparametric methods

In many modeling problems, linear methods are not enough to catch the complexity of the phenomenon which is, per se, nonlinear. In these situations, nonparametric are often more suited to obtain accurate approximations of the phenomenon under study. In this section, six nonparametric approaches are described: they are compared in Section 4 to model N\textsubscript{2}O fluxes and N leaching.

3.2.1. ACOSSO

Among nonparametric estimation approach, the smoothing splines (Wahba, 1990; Gu, 2002) is one of the most famous and widely used. Recently, (Storlie et al., 2011) presented the ACOSSO, an adaptive approach based on the COSSO method (Lin and Zhang, 2006) which is in the same line as smoothing splines: it is described as “a new regularization method for simultaneous model fitting and variable selection in nonparametric regression models in the framework of smoothing spline ANOVA”. This method penalizes the sum of component norms, instead of the squared norm employed in the traditional smoothing spline method. More precisely, in splines meta-modeling, it is useful to consider the ANOVA decomposition of \( f \) into terms of increasing dimensionality:

\[
f(x) = f(x^1, x^2, \ldots, x^d) = f_0 + \sum_j f^{(j)} + \sum_{k>j} f^{(jk)} + \ldots + f^{(1,2,\ldots,d)}
\]

where \( x^j \) is the \( j \)-th explanatory variable and where each term is a function only of the factors in its index, i.e. \( f^{(j)} = f(x^j), f^{(jk)} = f(x^j, x^k) \) and so on. The terms \( f^{(j)} \) represent the additive part of the model \( f \), while all higher order terms \( f^{(jk)} \ldots f^{(1,2,\ldots,d)} \) are denoted as “interactions”. The simplest example of smoothing spline ANOVA model is the additive model where only \( (f^{(j)})_{j=0,\ldots,d} \) are used.

To estimate \( f \), we make the usual assumption that \( f \in \mathcal{H} \), where \( \mathcal{H} \) is a RKHS (Reproducing Kernel Hilbert Space) (Berlinet and Thomas-Agnan, 2004). The space \( \mathcal{H} \) can be written as an orthogonal decomposition \( \mathcal{H} = \{1\} \oplus \bigoplus_{j=1}^q \mathcal{H}_j \), where each \( \mathcal{H}_j \) is itself a RKHS, \( \oplus \) is the direct sum of Hilbert spaces and \( j = 1, \ldots, q \) spans ANOVA terms of various orders. Typically \( q \) includes the main effects plus relevant interaction terms. \( f \) is then estimated by \( \hat{f} \) that minimizes a criterion being a trade-off between accuracy to the data (empirical mean squared error) and a penalty which
aims at minimizing each ANOVA term:

\[
\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{f}(x_i))^2 + \lambda_0 \sum_{j=1}^{q} \frac{1}{\theta_j} \| P^j \hat{f} \|_{\mathcal{H}}^2
\]

where \( P^j \hat{f} \) is the orthogonal projection of \( \hat{f} \) onto \( \mathcal{H}_j \) and the \( q \)-dimensional vector \( \theta_j \) of smoothing parameters needs to be tuned somehow, in such a way that each ANOVA component has the most appropriate degree of smoothness.

This statistical estimation problem requires the tuning of the \( d \) hyperparameters \( \theta_j \) (\( \lambda_0/\theta_j \) are also denoted as smoothing parameters). Various ways of doing that are available in the literature, by applying generalized cross-validation (GCV), generalized maximum likelihood procedures (GML) and so on (Wahba, 1990; Gu, 2002). But, in Eq. (3), \( q \) is often large and the tuning of all \( \theta_j \) is a formidable problem, implying that in practice the problem is simplified by setting \( \theta_j \) to 1 for any \( j \) and only \( \lambda_0 \) is tuned. This simplification, however, strongly limits the flexibility of the smoothing spline model, possibly leading to poor estimates of the ANOVA components.

Problem (3) also poses the issue of selection of \( \mathcal{H}_j \) terms: this is tackled rather effectively within the COSSO/ACOSSO framework. The COSSO (Lin and Zhang, 2006) penalizes the sum of norms, using a LASSO type penalty (Tibshirani, 1996) for the ANOVA model: LASSO penalties are \( L_1 \) penalties that lead to sparse parameters (i.e., parameters whose coordinates are all equal to zero except for a few ones). Hence, using this kind of penalties allows us to automatically select the most informative predictor terms \( \mathcal{H}_j \) with an estimate of \( \hat{f} \) that minimizes

\[
\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{f}(x_i))^2 + \lambda \sum_{j=1}^{Q} \| P^j \hat{f} \|_{\mathcal{H}}
\]

using a single smoothing parameter \( \lambda \), and where \( Q \) includes all ANOVA terms to be potentially included in \( \hat{f} \), e.g. with a truncation at 2\textsuperscript{nd} or 3\textsuperscript{rd} order interactions.

It can be shown that the COSSO estimate is also the minimizer of

\[
\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{f}(x_i))^2 + \sum_{j=1}^{Q} \frac{1}{\theta_j^j} \| P^j \hat{f} \|_{\mathcal{H}}^2
\]
subject to \( \sum_{j=1}^{Q} 1/\theta_j < M \) (where there is a 1-1 mapping between \( M \) and \( \lambda \)). So we can think of the COSSO penalty as the traditional smoothing spline penalty plus a penalty on the \( Q \) smoothing parameters used for each component. This can also be framed into a linear-quadratic problem, i.e. a quadratic objective (5) plus a linear constraint on \( 1/\theta_j \). The LASSO type penalty has the effect of setting some of the functional components (\( H_j \)'s) equal to zero (e.g. some variables \( x^j \) and some interactions (\( x^j, x^k \)) are not included in the expression of \( \hat{f} \)). Thus it “automatically” selects the appropriate subset \( q \) of terms out of the \( Q \) “candidates”. The key property of COSSO is that with one single smoothing parameter (\( \lambda \) or \( M \)) it provides estimates of all \( \theta_j \) parameters in one shot: therefore it improves considerably the simplified problem (3) by setting \( \theta_j = 1 \) (still with one single smoothing parameter \( \lambda_0 \)) and is much more computationally efficient than the full problem (3) with optimized \( \theta_j \)'s. An additional improvement from the COSSO is that the single smoothing parameter \( \lambda \) can be tuned to minimize the BIC (Bayesian Information Criterion) (Schwarz, 1978), thus allowing to target the most appropriate degree of parsimony of the metamodel. This is done by a simple grid-search algorithm as follows (see (Lin and Zhang, 2006) for details):

1. for each trial \( \lambda \) value, the COSSO estimate provides the corresponding values for \( \theta_j \) and subsequently its BIC;
2. the grid-search algorithm will provide the \( \hat{\lambda} \) with the smallest BIC.

The adaptive COSSO (ACOSSO) of (Storlie et al., 2011) is an improvement of the COSSO method: in ACOSSO, \( \hat{f} \in \mathcal{H} \) minimizes

\[
\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{f}(x_i))^2 + \lambda \sum_{j=1}^{q} w_j \| P^j \hat{f}(0) \|_{L^2} \tag{6}
\]

where \( 0 < w_j \leq \infty \) are weights that depend on an initial estimate, \( \hat{f}^{(0)} \), of \( f \), either using (3) with \( \theta_j = 1 \) or the COSSO estimate (4). The adaptive weights are obtained as \( w_j = \| P^j \hat{f}^{(0)} \|_{L^2}^{-\gamma} \), typically with \( \gamma = 2 \) and the \( L_2 \) norm \( \| P^j \hat{f}^{(0)} \|_{L^2} = (\int (P^j \hat{f}^{(0)}(x))^2 dx)^{1/2} \). The use of adaptive weights improves the predictive capability of ANOVA models with respect to the COSSO case: in fact it allows for more flexibility in estimating important functional components while giving a heavier penalty to unimportant functional components. The R scripts for ACOSSO can be found
at http://www.stat.lanl.gov/staff/CurtStorlie/index.html. In the present paper we used a MATLAB translation of such R script. The algorithm for tuning the hyper-parameters is then modified as follows:

1. an initial estimate of the ANOVA model \( \hat{f}^{(0)} \) is obtained either using (3) with \( \theta_j = 1 \) or the COSSO estimate (4);
2. given this trial ANOVA model \( \hat{f}^{(0)} \), the weights are computed as \( w_j = \|P_j \hat{f}^{(0)}\|^{-\gamma}_2 \);
3. given \( w_j \) and for each trial \( \lambda \) value, the ACOSSO estimate (6) provides the corresponding values for \( \theta_j \) and subsequently its BIC;
4. the grid-search algorithm will provide the \( \hat{\lambda} \) with the smallest BIC.

3.2.2. SDR-ACOSSO

In a “parallel” stream of research with respect to COSSO-ACOSSO, using the state-dependent parameter regression (SDR) approach of (Young, 2001), (Ratto et al., 2007) have developed a non-parametric approach, very similar to smoothing splines and kernel regression methods, based on recursive filtering and smoothing estimation (the Kalman filter combined with “fixed interval smoothing”). Such a recursive least-squares implementation has some key characteristics: (a) it is combined with optimal maximum likelihood estimation, thus allowing for an estimation of the smoothing hyper-parameters based on the estimation of a quality criterion rather than on cross-validation and (b) it provides greater flexibility in adapting to local discontinuities, heavy non-linearity and heteroscedastic error terms. Recently, (Ratto and Pagano, 2010) proposed a unified approach to smoothing spline ANOVA models that combines the best of SDR and ACOSSO: the use of the recursive algorithms in particular can be very effective in identifying the important functional components and in providing good estimates of the weights \( w_j \) to be used in (6), adding valuable information in the ACOSSO framework and allowing in many cases to improving ACOSSO performance. The Matlab script for this method can be found at http://eemc.jrc.ec.europa.eu/Software-SS_ANOVA_R.htm.

We summarize here the key features of Young’s recursive algorithms of SDR, by considering the case of \( d = 1 \) and \( f(x^1) = f^{(1)}(x^1) + e \), with \( e \sim N(0,\sigma^2) \). To do so, we rewrite the smoothing problem as \( y_i = s_i^1 + e_i \), where \( i = 1, \ldots, N \) and \( s_i^1 \) is the estimate of \( f^{(1)}(x_i^1) \). To make the recursive approach meaningful, the MC sample needs to be sorted in ascending order.
with respect to \( x^1 \): i.e. \( i \) and \( i - 1 \) subscripts are adjacent elements under such ordering, implying \( x^1_1 < x^1_2 < \ldots < x^1_i < \ldots < x^1_N \).

To recursively estimate the \( s^1_i \) in SDR it is necessary to characterize it in some stochastic manner, borrowing from non-stationary time series processes (Young and Ng, 1989; Ng and Young, 1990). In the present context, the integrated random walk (IRW) process provides the same smoothing properties of a cubic spline, in the overall State-Space formulation:

Observation Equation: \( y_i = s^1_i + e_i \)

State Equations: \[
\begin{align*}
s^1_i &= s^1_{i-1} + d^1_{i-1} \\
d^1_i &= d^1_{i-1} + \eta^1_i
\end{align*}
\] (7)

where \( d^1_i \) is the “slope” of \( s^1_i \), \( \eta^1_i \sim N(0, \sigma^2_{\eta^1}) \) and \( \eta^1_i \) (“system disturbance” in systems terminology) is assumed to be independent of the “observation noise” \( e_i \sim N(0, \sigma^2) \).

Given the ascending ordering of the MC sample, \( s^1_i \) can be estimated by using the recursive Kalman Filter (KF) and the associated recursive Fixed Interval Smoothing (FIS) algorithm (see e.g. (Kalman, 1960; Young, 1999) for details). First, it is necessary to optimize the hyper-parameter associated with the state space model (7), namely the Noise Variance Ratio (NVR), where \( \text{NVR}_1 = \sigma^2_{\eta^1}/\sigma^2 \). This is accomplished by maximum likelihood optimization (ML) using prediction error decomposition (Schwepp, 1965). The NVR plays the inverse role of a smoothing parameter: the smaller the NVR, the smoother the estimate of \( s^1_i \). Given the NVR, the FIS algorithm then yields an estimate \( \hat{s}^1_{i|N} \) of \( s^1_i \) at each data sample and it can be seen that the \( \hat{s}^1_{i|N} \) from the IRW process is the equivalent of \( \hat{f}^{(1)}(x^1_i) \) in the cubic smoothing spline model. At the same time, the recursive procedures provide, in a natural way, standard errors of the estimated \( \hat{s}^1_{i|N} \), that allow for the testing of their relative significance. Finally, it can be easily verified (Ratto and Pagano, 2010) that by setting \( \lambda/\theta_1 = 1/\text{NVR}_1 \cdot N^4 \), and with evenly spaced \( x^1_i \) values, the \( \hat{f}^{(1)}(x^1_i) \) estimate in the cubic smoothing spline model equals the \( \hat{s}^1_{i|N} \) estimate from the IRW process.

The most interesting aspect of the SDR approach is that it is not limited to the univariate case, but can be effectively extended to the most relevant multivariate one. In the general additive case, for example, the recursive procedure needs to be applied, in turn, for each term \( f^{(j)}(x^j_i) = \hat{s}^j_{i|N} \), requiring a different sorting strategy for each \( \hat{s}^j_{i|N} \). Hence the “back-fitting” procedure is applied, as described in (Young, 2000) and (Young, 2001). This procedure
provides both ML estimates of all NVR\(_j\)'s and the smoothed estimates of the additive terms \(\hat{s}_{i|N}\). So, the estimated NVR\(_j\)'s can be converted into \(\lambda_0/\theta_j\) values using \(\lambda_0/\theta_j = 1/(\text{NVR}_j \cdot N^4)\), allowing us to put the additive model into the standard cubic spline form.

In the SDR context, (Ratto and Pagano, 2010) formalized an interaction function as the product of two states \(s_1 \cdot s_2\), each of them characterized by an IRW stochastic process. Hence the estimation of a single interaction term \(f(x_i) = f^{(12)}(x_{i1}, x_{i2}) + e_i\) is expressed as:

\[
\text{Observation Equation: } y_i^* = s_{I1,i} \cdot s_{I2,i} + e_i \\
\text{State Equations: } (j = 1, 2) s_{j,i}^{I} = s_{j,i}^{I-1} + d_{j,i}^{I-1} \\
d_{j,i}^{I} = d_{j,i}^{I-1} + \eta_{j,i}^{I}
\]

where \(y_i^*\) is the model output after having taken out the main effects, \(I = 1, 2\) is the multi-index denoting the interaction term under estimation and \(\eta_{j,i}^{I} \sim N(0, \sigma_{j,i}^2)\). The two terms \(s_{j,i}^{I}\) are estimated iteratively by running the recursive procedure in turn.

The SDR recursive algorithms are usually very efficient in identifying the most appropriate way each ANOVA component individually, hence (Ratto and Pagano, 2010) proposed to exploit this in the ACOSSO framework as follows.

We define \(K_{(j)}\) to be the reproducing kernel (r.k.) of an additive term \(F_j\) of the ANOVA decomposition of the space \(F\). In the cubic spline case, this is constructed as the sum of two terms \(K_{(j)} = K_{01(j)} \oplus K_{1(j)}\) where \(K_{01(j)}\) is the r.k. of the parametric (linear) part and \(K_{1(j)}\) is the r.k. of the purely non-parametric part. The second order interaction terms are constructed as the tensor product of the first order terms, for a total of four elements, i.e.

\[
K_{(i,j)} = (K_{01(i)} \oplus K_{1(i)}) \otimes (K_{01(j)} \oplus K_{1(j)}) \\
= (K_{01(i)} \otimes K_{01(j)}) + (K_{01(i)} \otimes K_{1(j)}) + (K_{1(i)} \otimes K_{01(j)}) + (K_{1(i)} \otimes K_{1(j)})
\]

This suggested that a natural use of the SDR identification and estimation in the ACOSSO framework is to apply specific weights to each element of the r.k. \(K_{(i,j)}\) in (9). In particular the weights are the \(L_2\) norms of each of the four elements estimated in (8):

\[
\hat{s}_i^I \cdot \hat{s}_j^I = \hat{s}_{01(i)}^I \hat{s}_{01(j)}^I + \hat{s}_{1(i)}^I \hat{s}_{01(j)}^I + \hat{s}_{1(i)}^I \hat{s}_{01(j)}^I + \hat{s}_{1(i)}^I \hat{s}_{1(j)}^I
\]

As shown in (Ratto and Pagano, 2010), this choice can lead to a significant improvement in the accuracy of ANOVA models with respect to the original
ACOSSO approach. Overall, the algorithm for tuning the hyper-parameters in the combined SDR-ACOSSO reads:

1. the recursive SDR algorithm is applied to get an initial estimate of each ANOVA term in turn (back-fitting algorithm);
2. the weights are computed as the $L_2$ norms of the parametric and non-parametric parts of the cubic splines estimates;
3. given $w_j$ and for each trial $\lambda$ value, the ACOSSO estimate (6) provides the corresponding values for $\theta_j$ and subsequently its BIC;
4. the grid-search algorithm will provide the $\hat{\lambda}$ with the smallest BIC.

3.2.3. Kriging metamodel: DACE

DACE (Lophaven et al., 2002) is a Matlab toolbox used to construct kriging approximation models on the basis of data coming from computer experiments. Once we have this approximate model, we can use it as a meta-model (emulator, surrogate model). We briefly highlight the main features of DACE. The kriging model can be expressed as a regression

$$\hat{f}(x) = \beta_1 \phi^1(x) + \cdots + \beta_q \phi^q(x) + \zeta(x)$$  \hspace{1cm} (11)

where $\phi^j, j = 1, \ldots, q$ are deterministic regression terms (constant, linear, quadratic, etc.), $\beta_j$ are the related regression coefficients and $\zeta$ is a zero mean random process whose variance depends on the process variance $\omega^2$ and on the correlation $R(v, w)$ between $\zeta(v)$ and $\zeta(w)$. In kriging, correlation functions are typically used, defined as:

$$R(\theta, v - w) = \prod_{j=1}^{d} R_j(\theta_j, w_j - v_j).$$

In particular, for the generalized exponential correlation function, used in the present paper, one has

$$R_j(\theta_j, w_j - v_j) = \exp(-\theta_j |w_j - v_j|^{\theta_{d+1}})$$

Then, we can define $R$ as the correlation matrix at the training points (i.e., the matrix with coordinates $r_{i,j} = R(\theta, x_i, x_j)$) and the vector $r_x = [R(\theta, x_1, x), \ldots, R(\theta, x_N, x)]$, $x$ being an untried point. Similarly, we define the vector $\phi_x = [\phi^1(x) \cdots \phi^q(x)]^T$ and the matrix $\Phi = [\phi_{x_1} \cdots \phi_{x_N}]^T$ (i.e.,
\( \Phi \) stacks in matrix form all values of \( \phi_x \) at the training points. Then, considering the linear regression problem \( \Phi \beta \approx y \) coming from Eq. (11), with parameter \( \beta = [\beta_1, \ldots, \beta_q]^T \in \mathbb{R}^q \), the GLS solution is given by:

\[
\beta^* = (\Phi^T R^{-1} \Phi)^{-1} \Phi^T R^{-1} y
\]

which gives the predictor at untried \( x \)

\[
\hat{f}(x) = \phi_x^T \beta^* + r_x^T \gamma^*,
\]

where \( \gamma^* \) is the \( N \)-dimensional vector computed as \( \gamma^* = R^{-1} (y - \Phi \beta^*) \).

The proper estimation of the kriging metamodel requires, of course, to optimize the hyper-parameters \( \theta \) in the correlation function: this is typically performed by maximum likelihood. It is easy to check that the kriging predictor interpolates \( x_j \), if the latter is a training point.

It seems useful to underline that one major difference between DACE and ANOVA smoothing is the absence of any “observation error” in (11). This is a natural choice when analyzing computer experiments and it aims to exploit the “zero-uncertainty” feature of this kind of data. This, in principle, makes the estimation of kriging metamodels very efficient, as confirmed by the many successful applications described in literature and justifies the great success of this kind of metamodels among practitioners. It also seems interesting to mention the so-called “nugget” effect, which is also used in the kriging literature (Montes, 1994; Kleijnen, 2009). This is nothing other than a “small” error term in (11) and it often reduces some numerical problems encountered in the estimation of the kriging metamodels to the form of (11). The addition of a nugget term leads to kriging metamodels that smooth, rather than interpolate, making them more similar to other metamodels presented here.

### 3.2.4. Multilayer perceptron

“Neural network” is a general name for statistical methods dedicated to data mining. They comprise of a combination of simple computational elements (neurons or nodes) densely interconnected through synapses. The number and organization of the neurons and synapses define the network topology. One of the most popular neural network class is the “multilayer perceptrons” (MLP) commonly used to solve a wide range of classification and regression problems. In particular, MLP are known to be able to approximate any (smooth enough) complex function (Hornik, 1991). Perceptrons
were introduced at the end of the 50s by Rosenblatt but they started becoming very appealing more recently thanks to the soaring computational capacities of computers. The works of (Ripley, 1994) and (Bishop, 1995) provide a general description of these methods and their properties.

For the experiments presented in Section 4.3, one-hidden-layer perceptrons were used. They can be expressed as a function of the form

\[ f_w : \mathbf{x} \in \mathbb{R}^p \rightarrow g_1 \left( \sum_{i=1}^{Q} w_i^{(2)} g_2 \left( \mathbf{x}^T \mathbf{w}_i^{(1)} + w_i^{(0)} \right) + w_0^{(2)} \right) \]

where:

- \( w := \left[ (w_i^{(0)})_i, ((\mathbf{w}_i^{(1)})^T)_i, w_0^{(2)}, (w_i^{(2)})_i \right]^T \) are parameters of the model, called weights. They have to be learned in \((\mathbb{R})^Q \times (\mathbb{R}^p)^Q \times \mathbb{R} \times (\mathbb{R})^Q\) during the training;

- \( Q \) is a hyper-parameter indicating the number of neurons on the hidden layer;

- \( g_1 \) and \( g_2 \) are the activation functions of the neural networks. Generally, in regression cases (when the outputs to be predicted are real values rather than classes), \( g_1 \) is the identity function (hence the outputs are a linear combination of the neurons on the hidden layer) and \( g_2 \) is the logistic activation function \( z \rightarrow \frac{e^z}{1+e^z} \).

The weights are learned in order to minimize the mean square error on the training set:

\[ \mathbf{\hat{w}} := \text{arg min} \sum_{i=1}^{n} \| y_i - f_w(\mathbf{x}_i) \|^2. \] (12)

Unfortunately this error is not a quadratic function of \( w \) and thus no exact algorithm is available to find the global minimum of this optimization problem (and the existence of such a global minimum is not even guaranteed). Gradient descent based approximation algorithms are usually computed to find an approximate solution, where the gradient of \( w \rightarrow f_w(\mathbf{x}_i) \) is calculated by the back-propagation principle (Werbos, 1974).

Moreover, to avoid overfitting, a penalization strategy, called weight decay (Krogh and Hertz, 1992), was introduced. It consists of replacing the
minimization problem (12) by its penalized version:

\[ \hat{w} := \arg \min w \sum_{i=1}^{n} \|y_i - f_w(x_i)\|^2 + C\|w\|^2 \]

where \( C \) is the penalization parameter. The solution of this penalized mean square error is designed to be smoother than that given by Eq. (12). The \texttt{nnet} \texttt{R} function, provided in the \texttt{R} package \texttt{nnet} (Venables and Ripley, 2002), was used to train and test the one-hidden-layer MLP. As described in Section 4.3, a single validation approach was used to tune the hyper-parameters \( Q \) and \( C \) which were selected on a grid search (\( Q \in \{10, 15, 20, 25, 30\} \) and \( C \in \{0, 0.1, 1, 5, 10\} \)).

3.2.5. SVM (Support Vector Machines)

SVM were introduced by (Boser et al., 1992) originally to address classification problems. Subsequently (Vapnik, 1995) presented an application to regression problems to predict dependent real valued variables from given inputs. In SVM, the estimate \( \hat{f} \) is chosen among the family of functions

\[ f : x \in \mathbb{R}^d \rightarrow \langle w, \phi(x) \rangle_H + b \]

where \( \phi \) is a function from \( \mathbb{R}^d \) into a given Hilbert space \( (H, \langle ., . \rangle_H) \), here a RKHS, \( w \in H \) and \( b \in \mathbb{R} \) are parameters to be learned from the training dataset. Despite several strategies were developed to learn the parameters \( w \) and \( b \) (Steinwart and Christmann, 2008), we opted for the original approach which consists of using the \( \epsilon \)-insensitive loss function as a quality criterion for the regression:

\[ L_\epsilon(X, y, \hat{f}) = \sum_{i=1}^{N} \max \left( |\hat{f}(x_i) - y_i| - \epsilon, 0 \right). \]

This loss function has the property to avoid considering the error when it is small enough (smaller than \( \epsilon \)). His main interest, compared to the usual squared error, is its robustness (see (Steinwart and Christman, 2008) for a discussion). The SVM regression is based on the minimization of this loss function on the learning sample while penalizing the complexity of the obtained \( \hat{f} \). More precisely, the idea of SVM regression is to find \( w \) and \( b \) solutions of:

\[ \arg \min_{w,b} L_\epsilon(X, y, \hat{f}) + \frac{1}{C}\|w\|^2_H \]  

(13)
where the term $\|w\|^2_H$ is the regularization term that controls the complexity of $\hat{f}$ and $C$ is the regularization parameter: when $C$ is small, $\hat{f}$ is allowed to make bigger errors in favor of a smaller complexity; if the value of $C$ is high, $\hat{f}$ makes (almost) no error on the training data but it could have a large complexity and thus not be able to give good estimations for new observations (e.g., those of the test set). A good choice must devise a compromise between the accuracy required by the project and an acceptable metamodel complexity.

(Vapnik, 1995) demonstrates that, using the Lagrangian and Karush-Kuhn-Tucker conditions, $w$ takes the form

$$w = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \phi(x_i)$$

where $\alpha_i$ and $\alpha_i^*$ solve the so-called dual optimization problem:

$$\arg \max_{\alpha_i, \alpha_i^*} \left( -\frac{1}{2} \sum_{i,j=1}^{N} (\alpha_i - \alpha_i^*) (\alpha_i - \alpha_i^*) \langle \phi(x_i), \phi(x_j) \rangle_H - \epsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{N} y_i (\alpha_i - \alpha_i^*) \right)$$

subject to: $\sum_{i=1}^{N} (\alpha_i - \alpha_i^*) = 0$ and $\alpha_i, \alpha_i^* \in [0, C]$.

This is a classical quadratic optimization problem that can be explicitly solved. (Keerthi et al., 2001) provide a detailed discussion on the way to compute $b$ once $w$ is found; for the sake of clarity, in this paper we skip the full description of this step.

In Eq. (14), $\phi$ is only used through the dot products $\langle \phi(x_i), \phi(x_j) \rangle_H$. Hence, $\phi$ is never explicitly given but only accessed through the dot product by defining a kernel, $\mathcal{K}$:

$$\mathcal{K}(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_H.$$ (15)

This is the so-called kernel trick. As long as $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is symmetric and positive, it is ensured that an underlying Hilbert space $\mathcal{H}$ and an underlying $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$ exist satisfying the relation of Eq. (15). The very
common Gaussian kernel, \( K_\gamma(u,v) = e^{-\gamma \|u-v\|^2} \) for a \( \gamma > 0 \), was used in the simulations.

Finally, three hyper-parameters have to be tuned to use SVM regression:

- \( \epsilon \) of the loss function;
- \( C \), the regularization parameter of the SVM;
- \( \gamma \), the parameter of the Gaussian kernel.

As described in Section 4.3, a single validation approach was used to tune the hyper-parameters \( C \) and \( \gamma \) which were selected on a grid search \( (C \in \{10, 100, 1000, 2000\} \) and \( \gamma \in \{0.001, 0.01, 0.1\}) \). To reduce the computational costs and also to limit the number of hyperparameters to the same value as in MLP case (and thus to prevent the global method from being too flexible), we avoided tuning \( \epsilon \) by setting it equal to 1, which corresponds approximately to the second decile of the target variable for each scenario and output. This choice fitted the standard proposed by (Mattera and Haykin, 1998) which suggests having a number of Support Vectors smaller than 50% of the training set. Simulations were done by using the function \texttt{svm} from the \texttt{R} package \texttt{e1071} based on the libsvm library (Chang and Lin, 2001).

3.2.6. Random Forest

Random forests (RF) were first introduced by (Breiman, 2001) on the basis of his studies on bagging and of the works of (Amit and Geman, 1997; Ho, 1998) on features selection. Basically, bagging consists of computing a large number of elementary regression functions and of averaging them. In random forest, elementary regression functions involved in the bagging procedure are regression trees (Breiman et al., 1984). Building a regression tree aims at finding a series of splits deduced from one of the \( d \) variables, \( x^k \) (for a \( k \in \{1, \ldots, d\} \)), and a threshold, \( \tau \), that divides the training set into two subsamples, called nodes: \( \{i : x^k_i < \tau\} \) and \( \{i : x^k_i \geq \tau\} \). The split of a given node, \( \mathcal{N} \), is chosen, among all the possible splits, by minimizing the sum of the homogeneity of the two corresponding child nodes, \( \mathcal{N}^1_c \) and \( \mathcal{N}^2_c \), as follows:

\[
\sum_{i \in \mathcal{N}^1_c} \left( y_i - \bar{y}^{\mathcal{N}^1_c} \right)^2
\]

where \( \bar{y}^{\mathcal{N}^1_c} = \frac{1}{|\mathcal{N}^1_c|} \sum_{i \in \mathcal{N}^1_c} y_i \) is the mean value of the output variable for the observations belonging to \( \mathcal{N}^1_c \) (i.e., the intra-node variance).
The growth of the tree stops when the child nodes are homogeneous enough (for a previously fixed value of homogeneity) or when the number of observations in the child nodes is smaller than a fixed number (generally chosen between 1 and 5). The prediction obtained for new inputs, \( \mathbf{x} \), is then simply the mean of the outputs, \( y_i \), of the training set that belong to the same terminal node (a leaf). The pros of this method are its easy readability and interpretability; the main drawback is its limited flexibility, especially for regression problems. To overcome this limit, random forests combine a large number (several hundreds or several thousands) of regression trees, \( T \).

In the forest, each tree is built sticking to the following algorithm that is made of random perturbations of the original procedure to make the tree under-efficient (i.e., so that none of the tree in the forest is the optimal one for the training dataset):

1. A given number of observations, \( m \), are randomly chosen from the training set: this subset is called *in-bag* sample whereas the other observations are called *out-of-bag* and are used to check the error of the tree;
2. For each node of the tree, a given number of variables, \( q \), are randomly selected among all the possible explanatory variables. The best split is then calculated on the basis of these \( q \) variables for the \( m \) chosen observations.

All trees in the forest are fully learned: the final leafs all have homogeneity equal to 0. Once having defined the \( T \) regression trees, \( T_1, \ldots, T_T \), the regression forest prediction for new input variables, \( \mathbf{x} \), is equal to the mean of the individual predictions obtained by each tree of the forest for \( \mathbf{x} \).

Several hyper-parameters can be tuned for random forests such as the number of trees in the final forest or the number of variables randomly selected to build a given split. But, as this method is less sensitive to parameter tuning than the other ones (i.e., SVM and MLP), we opted for leaving the default values implemented in the \texttt{R} package \texttt{randomForest} based on useful heuristics: 500 trees were trained, each defined from a bootstrap sample built with replacement and having the size of the original dataset. Each node was defined from three randomly chosen variables and the trees were grown until the number of observations in each node was smaller than five. Moreover, the full learning process always led to a stabilized out-of-bag error.
4. Simulations and results

4.1. Application to the Cross Compliance Assessment Tool

As described above in the Section 1, the impact assessment of Cross Compliance (CC) measures on the EU27 farmlands, required the development of a simulation platform called Cross Compliance Assessment Tool (CCAT). The CCAT framework integrates different models, such as Miterra (Velthof et al., 2009), DNDC-EUROPE (Follador et al., 2011), EPIC (van der Velde et al., 2009) and CAPRI (Britz and Witzke, 2008; Britz, 2008), in order to guarantee an exhaustive evaluation of the effects of agro-environmental standards for different input, scenario assumptions, compliance rates and space-time resolutions (Elbersen et al., 2010; De Vries et al., 2008). The simulated outputs are indicators for nitrogen (N) and carbon (C) fluxes, biodiversity and landscape, market response and animal welfare. The selection of the CC scenarios as well as the definition of the environmental indicators to be considered in this project, are described by (Jongeneel et al., 2008). The CCAT tool evaluates the effect of agricultural measures on N$_2$O fluxes and N leaching by means of the meta-model of the mechanistic model DNDC-EUROPE (Follador et al., 2011). N$_2$O is an important greenhouse gas (Intergovernmental Panel on Climate Change, 2007). Agriculture and in particular agricultural soils are contributing significantly to anthropogenic N$_2$O emissions (European Environment Agency, 2010). N$_2$O fluxes from soils are characterized by a high spatial variability and the accuracy of estimates can be increased if spatially explicit information is taken into consideration (Leip et al., 2011a). Similarly, leaching of nitrogen from agricultural soils is an important source of surface and groundwater pollution (European Environment Agency, 1995).

The main limits of using DNDC-EUROPE directly in the CCAT platform are the high computational costs and memory requirements, due to the large size of input datasets and the complexity and high number of equations to solve. To mitigate this problem, making the integration easier, we decided to develop a metamodel of DNDC-EUROPE (Follador and Leip, 2009). The choice of the best meta-modeling approach has been based on the analysis of performance of different algorithms, as described in details in Section 4.4. The best metamodel is expected to have low computational costs and an acceptable accuracy for all the dataset sizes.
4.2. Input and Output data description

The set of training observations (around 19,000 observations) used to define a metamodel \( \hat{f} \) was created by linking the agro-economic CAPRI dataset with the bio-geochemical DNDC-EUROPE model at Homogeneous Spatial Mapping Unit (HSMU) resolution, as described in (Leip et al., 2008). We opted for corn cultivation as case study, since it covers almost 4.6% of UAA (utilized agricultural area) in EU27, playing an important role in human and animal food supply (European Union Commission, 2010)\(^1\) and representing one of the main cropping systems in Europe. To obtain a representative sample of situations for the cultivation of corn in EU27, we selected about 19,000 HSMUs on which at least 10% of the agricultural land was used for corn (Follador et al., 2011).

The input observations used to train the metamodels were drawn from the whole DNDC-EUROPE input database (Leip et al., 2008; Li et al., 1992; Li, 2000), in order to meet the need of simplifying the I/O flux of information between models in the CCAT platform. This screening was based on a preliminary sensitivity analysis of input data through the importance function of the \texttt{R} package \texttt{randomForest}, and subsequently it was refined by expert evaluations (Follador et al., 2011; Follador and Leip, 2009). At last, 11 input variables were used:

- Variable related to N input [kgN ha\(^{-1}\)yr\(^{-1}\)], such as mineral fertilizer (N\(_{\text{FR}}\)) and manure (N\(_{\text{MR}}\)) amendments, N from biological fixation (N\(_{\text{fix}}\)) and N in crop residue (N\(_{\text{res}}\));
- Variables related to soil: soil bulk density, BD, [g cm\(^{-3}\)], topsoil organic carbon, SOC, [mass fraction], clay content, clay, [fraction] and topsoil pH, pH;
- Variables related to climate: annual precipitation Rain, [mm yr\(^{-1}\)], annual temperature T\(_{\text{mean}}\) [°C] and N in rain, N\(_{\text{r}}\), [ppm].

They refer to the main driving forces taking part in the simulation of N\(_2\)O and N leaching with DNDC-EUROPE, such as farming practices, soil attributes and climate information. In this contribution we only show the results for the corn baseline scenario - that is the conventional corn cultivation in EU27,
as described by (Follador et al., 2011). Note that a single metamodel was developed for each CC scenario and for each simulated output in CCAT, as described in (Follador and Leip, 2009). Figure 1 summarizes the relations between the DNDC-EUROPE model and the metamodel.

As the number of input variables was not large, they were all used in all the metamodeling methods described in Section 3, without additional variable selection. The only exception is the second linear model (Section 3.1) which uses a more complete list of input variables obtained by various combinations of the original 11 variables and thus includes a variable selection process to avoid collinearity issues.

Two output variables were studied: the emissions of $N_2O$ ([kg N yr$^{-1}$ ha$^{-1}$] for each HSMU), a GHG whose reduction is a leading matter in climate change mitigation strategies, and the nitrogen leaching ([kg N yr$^{-1}$ ha$^{-1}$] for each HSMU), which has to be monitored to meet the drinking water quality standards (Askegaard et al., 2005). A metamodel was developed for each single output variable. The flux of information through the DNDC-EUROPE model and its relationship with the metamodel’s one are summarized in Figure 1. The data were extracted using a high performance computer cluster and the extraction process took more than one day for all the 19 000 observations.

4.3. Training, validation and test approach

The training observations were randomly partitioned (without replacement) into two groups: 80% of the observations (i.e., $N_L \simeq 15 000$ HSMU) were used for training (i.e., for defining a convenient $\hat{f}$) and the 20% remaining observations (i.e., $N_T \simeq 4 000$ HSMU) were used for validating the metamodels (i.e., for calculated an error score). Additionally, in order to understand the impact of the training dataset on the goodness of the estimations ($\hat{y}_i$) and to compare the different metamodel performance according to the data availability, we randomly selected from the entire training set a series of subsets, having respectively $N_L = 8 000, 4 000, 2 000, 1 000, 500, 200$ and $100$ observations, each consecutive training subset being a subset of the previous one.

The methodology used to assess the behavior of different metamodels under various experimental conditions (size of the dataset and nature of the output) are summarized in Description 1.

More precisely, for some metamodels, Step 2 requires the tuning of some hyper-parameters (e.g., SVM have three hyper-parameters, see Section 3).
Figure 1: Flow of data through the DNDC-EUROPE model (M) and relationship with the metamodel’s one (MM). The input variables of the metamodels were selected from the original DNDC-EUROPE dataset (screening). The estimated (*) output were compared with the detailed model’s output during the training and test phases to improve the metamodel and to evaluate the goodness of the approximation.
Description 1 Methodology used to compare the metamodels under various experimental conditions

1: for Each metamodel, each output and each size $N_L$ do
2:  
3:  
4:  
5: end for

These hyper-parameters were tuned by:

- for ACOSSO and SDR: a grid-search to minimize BIC plus an algorithm to get the weights $w_j$: in these cases, an efficient formula, that does not require to compute each leave-one-out estimate of $f$, can be used to compute the BIC; moreover the COSSO penalty provides all $\theta_j$ given $\lambda$ and $w_j$ in a single shot. In the SDR identification steps, a maximum likelihood strategy is applied to optimize NVR’s;

- for DACE, a maximum likelihood strategy;

- for MLP, SVM and RF, a \textit{simple validation strategy} preferred to a cross validation strategy to reduce the computational time especially with the largest training datasets): half of the data were used to define several metamodels depending on the values of hyper-parameters on a grid search and the remaining data were used to select the best set of hyper-parameters by minimizing a mean square error criterion.

Hence, depending on which features are the most interesting (easy tuning of the hyperparameters, size of the training dataset, size of the dataset needing new prediction...), the use of one method is more or less recommended. Table 1 summarizes the main characteristics of the training and validation steps of each method as well as the characteristics to do new predictions. For instance, linear models are more
Table 1: Summary of the main features for training, validation (hyperparameters tuning) and test steps of each method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training characteristics</th>
<th>Validation characteristics</th>
<th>New predictions characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM1</td>
<td>Very fast to train.</td>
<td>There is <strong>no hyperparameter</strong> to tune.</td>
<td>Very fast.</td>
</tr>
<tr>
<td>LM2</td>
<td><strong>Fast</strong> to train but much slower than LM1 because of the number of parameters to learn.</td>
<td>There is <strong>no hyperparameter</strong> to tune.</td>
<td>Very fast.</td>
</tr>
<tr>
<td>ACOSSO</td>
<td><strong>Fast to train only if the number of observations is very low:</strong> the dimension of the kernel matrix is $N_L \times N_L$ and it is obtained as the sum of the kernels of each $[N_L \times N_L]$ ANOVA term, which can be long to calculate.</td>
<td>One hyper-parameter ($\lambda$) is tuned twice by minimizing BIC: the first time to get the weights $w_j$ the second to get the final estimate (given $\lambda$ and $w_j$ the COSSO penalty provides automatically in a single shot all $\theta_j$).</td>
<td>The time needed to obtain new predictions <strong>can be high</strong> depending on the sizes of both the training dataset and the test dataset. It requires to compute a kernel matrix having dimension $N_L \times N_T$.</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Method</th>
<th>Training characteristics</th>
<th>Validation characteristics</th>
<th>Test characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDR</td>
<td><strong>Fast to train only if the number of observations is very low:</strong> the dimension of the kernel matrix is $N_L \times N_L$ and it is obtained as the sum of the kernels of each $[N_L \times N_L]$ ANOVA term, which can be long to calculate.</td>
<td>As for ACOSSO, the single hyperparameter ($\lambda$) is tuned by minimizing BIC: the SDR identification step to provide $w_j$ also optimizes hyperparameters for each ANOVA component but this can be done efficiently by the SDR recursive algorithms (given $\lambda$ and $w_j$ the COSSO penalty provides automatically in a single shot all $\theta_j$).</td>
<td>The time needed to obtain new predictions <strong>can be high</strong> depending on the sizes of both the training dataset and the test dataset. It requires to compute a kernel matrix having dimension $N_L \times N_T$.</td>
</tr>
<tr>
<td>DACE</td>
<td><strong>Fast to train only if the number of observations is very low:</strong> the dimension of the kernel matrix is $N_L \times N_L$, and the inversion of a matrix $N_L \times N_L$ is required in the GLS procedure.</td>
<td>$d + 1$ hyperparameters are tuned by ML, which becomes intractable already for moderate $d$: each step of the optimization a matrix $N_L \times N_L$ has to be inverted.</td>
<td>The time needed to obtain new predictions <strong>can be high</strong> depending on the sizes of both the training dataset and the test dataset. It requires to compute a kernel matrix having dimension $N_L \times N_T$.</td>
</tr>
</tbody>
</table>

*Continued on next page*
Table 1 – Continued from previous page

<table>
<thead>
<tr>
<th>Method</th>
<th>Training characteristics</th>
<th>Validation characteristics</th>
<th>Test characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td><strong>Hard to train</strong>: because the error to minimize is not quadratic, the training step faces local minima problems and has thus to be performed several times with various initialization values. It is also very sensitive to the dimensionality of the data (that strongly increases the number of weights to train) and, to a lesser extent, to the number of observations.</td>
<td>2 hyperparameters have to be tuned but one is discrete (number of neurons on the hidden layer) which is easier. Nervelessness, cross validation is not suited: tuning is performed by simple validation and can thus be less accurate. It can be time consuming.</td>
<td>The time needed to obtain new predictions is <strong>very low</strong>: it depends on the number of predictions.</td>
</tr>
<tr>
<td>SVM</td>
<td><strong>Fast to train if the number of observations is low</strong>: SVM are almost insensitive to the dimensionality of the data but the dimension of the kernel matrix is $N_L \times N_I$ and can be long to calculate.</td>
<td>Three hyperparameters have to be tuned and in the case where the size of the training dataset is large, cross validation is not suited. Tuning is performed by simple validation and can thus be less accurate. It is also time consuming.</td>
<td>The time needed to obtain new predictions can be high depending on the sizes of both the training dataset and the test dataset. It requires to compute a kernel matrix having dimension $N_L \times N_T$.</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1 – *Continued from previous page*

<table>
<thead>
<tr>
<th>Method</th>
<th>Training characteristics</th>
<th>Validation characteristics</th>
<th>Test characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>Fast to train: almost insensitive to the size or the dimensionality of the training dataset thanks to the random selections of observations and variables. Most of the time needed to train is due to the number of trees required to stabilize the algorithm, that can sometimes be large.</td>
<td>Almost insensitive to hyperparameters so <strong>no extensive tuning</strong> is required.</td>
<td>The time needed to obtain new predictions is <strong>low</strong>: it depends on the number of predictions to do and also on the number of trees in the forest.</td>
</tr>
</tbody>
</table>

In Step 4, the test quality criterion was evaluated by calculating several quantities:

- the *Mean Squared Error* (MSE):
  \[
  \text{MSE} = \frac{1}{NT} \sum_{i=1}^{NT} (\hat{y}_i - y_i)^2
  \]
  where \(y_i\) and \(\hat{y}_i\) are, respectively, the model outputs in the test dataset and the corresponding approximated outputs given by the metamodel.

- the \(R^2\) coefficient:
  \[
  R^2 = 1 - \frac{\sum_{i=1}^{NT} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{NT} (\hat{y}_i - \overline{y})^2} = 1 - \frac{\text{MSE}}{\text{Var}(y)}
  \]
  where \(\overline{y}\) and \(\text{Var}(y)\) are the mean and the variance of all \(y_i\) in the test dataset. \(R^2\) is equal to 1 if the predictions are perfect and thus gives a way to quantify the accuracy of the predictions to the variability of the variable to predict.
• the standard deviation of the SE and the maximum value of the SE were also computed to give an insight on the variability of the performance and not only on its mean.

4.4. Results and discussion

This section includes several ways to compare the methods on the problem described in 4.2. First, Section 4.4.1 compares the accuracy of the predictions for various methods and various training dataset sizes. Then, Section 4.4.2 gives a comparison of the computational times needed either to train the model (with the maximum dataset size) and to make new predictions. Finally, Section 4.4.3 describes the model itself and gives an insight about its physical interpretation.

4.4.1. Accuracy

The performance on the test set is summarized in Tables 2 to 5: they include characteristics about the mean values of the squared errors (MSE and $R^2$) in Tables 2 and 3, respectively for $N_2O$ and N leaching predictions, as well as characteristics related to the variability of the performance (standard deviations of the squared errors and maximum values of the squared errors) in Tables 4 and 5, respectively for $N_2O$ and N leaching predictions. Note that, in almost all cases, the minimum values of the squared errors were equal or close to 0.

The evolution of $R^2$ on the test set in function of the size of the training set is displayed in Figures 2 (N$_2$O prediction) and 3 (N leaching) for each method.

From these results, several facts clearly appeared:

• Even for small datasets, the metamodeling approach behaves correctly with $R^2$ always greater than 80% for the best approaches. Note that the poorest results (those that are the closest to 80%) are obtained for small training dataset sizes (100 or 200). This means that, in the case where several metamodels are needed to model various assumptions of the input variables ranges, crude but acceptable estimates can be obtained at a very low computational cost. For more efficient predictions, larger datasets are more suited and achieve $R^2$ values greater than 90%.

• Predicting N leaching seems an easier task than predicting $N_2O$ fluxes with greater performance for almost any training dataset size. This is not surprising because $N_2O$ is generated as an intermediate product in...
<table>
<thead>
<tr>
<th>Size of the dataset</th>
<th>LM1</th>
<th>LM2</th>
<th>Dace</th>
<th>SDR</th>
<th>Acosso</th>
<th>MLP</th>
<th>SVM</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>67.22%</td>
<td>✓</td>
<td>74.03%</td>
<td>78.50%</td>
<td>80.40%</td>
<td>58.68%</td>
<td>50.26%</td>
<td>49.90%</td>
</tr>
<tr>
<td></td>
<td>11.50</td>
<td>✓</td>
<td>9.11</td>
<td>7.54</td>
<td>6.88</td>
<td>14.50</td>
<td>17.45</td>
<td>17.57</td>
</tr>
<tr>
<td>200</td>
<td>66.91%</td>
<td>-13 093%</td>
<td>77.74%</td>
<td>81.50%</td>
<td>78.88%</td>
<td>65.86%</td>
<td>63.05%</td>
<td>51.87%</td>
</tr>
<tr>
<td></td>
<td>11.61</td>
<td>4 626</td>
<td>7.81</td>
<td>6.49</td>
<td>7.41</td>
<td>11.98</td>
<td>12.96</td>
<td>16.89</td>
</tr>
<tr>
<td>500</td>
<td>75.20%</td>
<td>-163%</td>
<td>83.07%</td>
<td>76.04%</td>
<td>78.39%</td>
<td>73.81%</td>
<td>83.86%</td>
<td>69.91%</td>
</tr>
<tr>
<td></td>
<td>8.70</td>
<td>92.35</td>
<td>5.94</td>
<td>8.41</td>
<td>7.58</td>
<td>9.19</td>
<td>5.66</td>
<td>10.56</td>
</tr>
<tr>
<td>1 000</td>
<td>76.85%</td>
<td>65.94%</td>
<td>85.58%</td>
<td>82.16%</td>
<td>77.60%</td>
<td>78.81%</td>
<td>84.62%</td>
<td>76.47%</td>
</tr>
<tr>
<td></td>
<td>8.47</td>
<td>11.95</td>
<td>5.06</td>
<td>6.26</td>
<td>7.86</td>
<td>7.69</td>
<td>5.40</td>
<td>8.25</td>
</tr>
<tr>
<td>2 000</td>
<td>76.89%</td>
<td>76.40%</td>
<td>81.34%</td>
<td>84.16%</td>
<td>78.26%</td>
<td>84.94%</td>
<td>85.73%</td>
<td>77.86%</td>
</tr>
<tr>
<td></td>
<td>8.11</td>
<td>8.28</td>
<td>6.55</td>
<td>5.27</td>
<td>7.63</td>
<td>5.28</td>
<td>5.01</td>
<td>7.77</td>
</tr>
<tr>
<td>4 000</td>
<td>77.24%</td>
<td>55.67%</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>88.91%</td>
<td>87.33%</td>
<td>86.01%</td>
</tr>
<tr>
<td></td>
<td>7.99</td>
<td>15.55</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3.89</td>
<td>4.45</td>
<td>4.90</td>
</tr>
<tr>
<td>8 000</td>
<td>77.05%</td>
<td>84.62%</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>88.85%</td>
<td>88.98%</td>
<td>89.89%</td>
</tr>
<tr>
<td></td>
<td>8.05</td>
<td>5.40</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3.91</td>
<td>3.86</td>
<td>3.55</td>
</tr>
<tr>
<td>≥ 15 000</td>
<td>77.10%</td>
<td>87.60%</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>90.66%</td>
<td>91.05%</td>
<td>92.29%</td>
</tr>
<tr>
<td></td>
<td>8.03</td>
<td>3.28</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3.28</td>
<td>3.14</td>
<td>2.71</td>
</tr>
</tbody>
</table>

Table 2: $R^2$ (first line) and MSE (second line) on the test set for each method and various sizes of the training dataset for $\text{N}_2\text{O}$ prediction. For each size, the best $R^2$ is in bold. ✓ corresponds to cases impossible to train, either because the model is over-specified (more parameters to estimate than the number of observations: LM2) or because the training size is too large for the method to be used (Dace/SDR/Acosso).
<table>
<thead>
<tr>
<th>Size of the dataset</th>
<th>LM1</th>
<th>LM2</th>
<th>Dace</th>
<th>SDR</th>
<th>Acosso</th>
<th>MLPL</th>
<th>SVM</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>67.57%</td>
<td>✓</td>
<td>79.46%</td>
<td>81.72%</td>
<td>79.69%</td>
<td>76.56%</td>
<td>73.54%</td>
<td>71.94%</td>
</tr>
<tr>
<td></td>
<td>1 742</td>
<td>✓</td>
<td>1 103</td>
<td>982</td>
<td>1 091</td>
<td>1 259</td>
<td>1 421</td>
<td>1 507</td>
</tr>
<tr>
<td>200</td>
<td>67.77%</td>
<td>-2 086%</td>
<td>83.49%</td>
<td>85.36%</td>
<td>86.08%</td>
<td>82.61%</td>
<td>84.06%</td>
<td>74.85%</td>
</tr>
<tr>
<td></td>
<td>1 731</td>
<td>&gt; 10^6</td>
<td>887</td>
<td>786</td>
<td>747</td>
<td>934</td>
<td>856</td>
<td>1 351</td>
</tr>
<tr>
<td>500</td>
<td>69.05%</td>
<td>36.92%</td>
<td>87.17%</td>
<td>86.20%</td>
<td>86.17%</td>
<td>83.69%</td>
<td>86.26%</td>
<td>78.51%</td>
</tr>
<tr>
<td></td>
<td>1 662</td>
<td>3 388</td>
<td>689</td>
<td>741</td>
<td>743</td>
<td>876</td>
<td>738</td>
<td>1 154</td>
</tr>
<tr>
<td>1 000</td>
<td>69.19%</td>
<td>27.24%</td>
<td>89.08%</td>
<td>88.43%</td>
<td>89.00%</td>
<td>85.13%</td>
<td>85.59%</td>
<td>83.44%</td>
</tr>
<tr>
<td></td>
<td>1 655</td>
<td>3 908</td>
<td>587</td>
<td>621</td>
<td>591</td>
<td>799</td>
<td>774</td>
<td>889</td>
</tr>
<tr>
<td>2 000</td>
<td>70.13%</td>
<td>60.62%</td>
<td>93.90%</td>
<td>91.39%</td>
<td>91.33%</td>
<td>84.94%</td>
<td>89.77%</td>
<td>85.07%</td>
</tr>
<tr>
<td></td>
<td>1 604</td>
<td>2 115</td>
<td>328</td>
<td>462</td>
<td>466</td>
<td>655</td>
<td>549</td>
<td>802</td>
</tr>
<tr>
<td>4 000</td>
<td>70.21%</td>
<td>89.92%</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>93.26%</td>
<td>87.33%</td>
</tr>
<tr>
<td></td>
<td>1 600</td>
<td>541</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>521</td>
<td>362</td>
</tr>
<tr>
<td>8 000</td>
<td>70.28%</td>
<td>90.78%</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>92.43%</td>
<td>95.49%</td>
</tr>
<tr>
<td></td>
<td>1 596</td>
<td>495</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>406</td>
<td>242</td>
</tr>
<tr>
<td>≃ 15 000</td>
<td>70.28%</td>
<td>91.52%</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>89.65%</td>
<td>96.65%</td>
</tr>
<tr>
<td></td>
<td>1 596</td>
<td>455</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>556</td>
<td>180</td>
</tr>
</tbody>
</table>

Table 3: $R^2$ (first line) and MSE (second line) on the test set for each method and various sizes of the training dataset for N leaching prediction. For each size, the best $R^2$ is in bold. ✓ corresponds to cases impossible to train, either because the model is over-specified (more parameters to estimate than the number of observations: LM2) or because the training size is too large for the method to be used (Dace/SDR/Acosso).
<table>
<thead>
<tr>
<th>Size of the dataset</th>
<th>LM1</th>
<th>LM2</th>
<th>Dace</th>
<th>SDR</th>
<th>Acosso</th>
<th>MLP</th>
<th>SVM</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>80.4</td>
<td>✓</td>
<td>72.7</td>
<td>52.4</td>
<td>50.2</td>
<td>125.5</td>
<td>159.6</td>
<td>150.0</td>
</tr>
<tr>
<td></td>
<td>2 400</td>
<td>✓</td>
<td>2 319</td>
<td>1 845</td>
<td>1 597</td>
<td>2 911</td>
<td>3 816</td>
<td>3 538</td>
</tr>
<tr>
<td>200</td>
<td>84.5</td>
<td>&gt; 10^5</td>
<td>68.1</td>
<td>52.3</td>
<td>64.6</td>
<td>100.3</td>
<td>113.7</td>
<td>145.4</td>
</tr>
<tr>
<td></td>
<td>2 461</td>
<td>&gt; 10^6</td>
<td>2 207</td>
<td>1 915</td>
<td>2 098</td>
<td>2 534</td>
<td>2 636</td>
<td>3 352</td>
</tr>
<tr>
<td>500</td>
<td>59.3</td>
<td>1 472.9</td>
<td>49.6</td>
<td>74.0</td>
<td>60.2</td>
<td>84.9</td>
<td>42.5</td>
<td>99.1</td>
</tr>
<tr>
<td></td>
<td>2 027</td>
<td>48 769</td>
<td>1 928</td>
<td>2 589</td>
<td>2 303</td>
<td>2 172</td>
<td>1 753</td>
<td>2 718</td>
</tr>
<tr>
<td>1 000</td>
<td>56.9</td>
<td>203.5</td>
<td>48.6</td>
<td>51.0</td>
<td>63.4</td>
<td>53.9</td>
<td>48.5</td>
<td>77.7</td>
</tr>
<tr>
<td></td>
<td>1 980</td>
<td>8 384</td>
<td>1 643</td>
<td>1 633</td>
<td>2 065</td>
<td>1 888</td>
<td>1 874</td>
<td>2 348</td>
</tr>
<tr>
<td>2 000</td>
<td>50.3</td>
<td>81.5</td>
<td>66.7</td>
<td>37.8</td>
<td>62.9</td>
<td>38.4</td>
<td>41.6</td>
<td>70.4</td>
</tr>
<tr>
<td></td>
<td>1 826</td>
<td>2 890</td>
<td>2 456</td>
<td>1 212</td>
<td>3 000</td>
<td>1 039</td>
<td>1 663</td>
<td>2 421</td>
</tr>
<tr>
<td>4 000</td>
<td>46.1</td>
<td>539.2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>33.0</td>
<td>37.6</td>
<td>52.8</td>
</tr>
<tr>
<td></td>
<td>1 711</td>
<td>32 290</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1 110</td>
<td>1 519</td>
<td>2 040</td>
</tr>
<tr>
<td>8 000</td>
<td>42.2</td>
<td>60.9</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>31.0</td>
<td>43.2</td>
<td>38.3</td>
</tr>
<tr>
<td></td>
<td>1 564</td>
<td>2 846</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1 072</td>
<td>1 773</td>
<td>1 645</td>
</tr>
<tr>
<td>≃ 15 000</td>
<td>42.2</td>
<td>29.0</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>29.0</td>
<td>35.7</td>
<td>25.6</td>
</tr>
<tr>
<td></td>
<td>1 568</td>
<td>1 339</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1 339</td>
<td>1 833</td>
<td>807</td>
</tr>
</tbody>
</table>

Table 4: Standard deviation (first line) and maximum (second line) of the squared errors on the test set for each method and various sizes of the training dataset for N$_2$O prediction. For each size, the minimal standard deviation and the minimal value of the maxima are in bold. ✓ corresponds to cases impossible to train, either because the model is over-specified (more parameters to estimate than the number of observations: LM2) or because the training size is too large for the method to be used (Dace/SDR/Acosso).
<table>
<thead>
<tr>
<th>Size of the dataset</th>
<th>LM1</th>
<th>LM2</th>
<th>Dace</th>
<th>SDR</th>
<th>Acosso</th>
<th>MLP</th>
<th>SVM</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>6.11</td>
<td>✓</td>
<td>5.83</td>
<td>6.45</td>
<td>8.14</td>
<td>6.72</td>
<td>9.79</td>
<td>7.99</td>
</tr>
<tr>
<td></td>
<td>173.1</td>
<td>✓</td>
<td>180.5</td>
<td>177.0</td>
<td>241.1</td>
<td>238.5</td>
<td>367.7</td>
<td>275.7</td>
</tr>
<tr>
<td>200</td>
<td>6.26</td>
<td>&gt; 10^4</td>
<td>5.24</td>
<td>7.15</td>
<td>8.61</td>
<td>6.28</td>
<td>5.36</td>
<td>7.95</td>
</tr>
<tr>
<td></td>
<td>184.7</td>
<td>&gt; 10^5</td>
<td>152.4</td>
<td>290.6</td>
<td>279.5</td>
<td>213.8</td>
<td>146.8</td>
<td>284.6</td>
</tr>
<tr>
<td>500</td>
<td>6.99</td>
<td>45.7</td>
<td>7.34</td>
<td>7.38</td>
<td>8.62</td>
<td>6.89</td>
<td>6.77</td>
<td>7.83</td>
</tr>
<tr>
<td></td>
<td>204.1</td>
<td>1 427.7</td>
<td>238.2</td>
<td>280.0</td>
<td>280.9</td>
<td>213.8</td>
<td>302.8</td>
<td>290.7</td>
</tr>
<tr>
<td>1 000</td>
<td>7.37</td>
<td>82.3</td>
<td>7.64</td>
<td>7.10</td>
<td>8.90</td>
<td>7.72</td>
<td>10.24</td>
<td>7.47</td>
</tr>
<tr>
<td></td>
<td>220.9</td>
<td>4 090.4</td>
<td>270.6</td>
<td>239.8</td>
<td>255.3</td>
<td>289.0</td>
<td>358.1</td>
<td>291.1</td>
</tr>
<tr>
<td>2 000</td>
<td>5.91</td>
<td>71.9</td>
<td>2.66</td>
<td>3.15</td>
<td>9.13</td>
<td>5.74</td>
<td>6.63</td>
<td>5.53</td>
</tr>
<tr>
<td></td>
<td>177.4</td>
<td>4 309.1</td>
<td>96.6</td>
<td>113.3</td>
<td>128.7</td>
<td>225.9</td>
<td>320.6</td>
<td>212.7</td>
</tr>
<tr>
<td>4 000</td>
<td>5.71</td>
<td>4.94</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3.50</td>
<td>3.61</td>
</tr>
<tr>
<td></td>
<td>167.0</td>
<td>213.5</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>134.5</td>
<td>123.1</td>
</tr>
<tr>
<td>8 000</td>
<td>5.59</td>
<td>4.31</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>2.80</td>
<td>2.38</td>
</tr>
<tr>
<td></td>
<td>162.0</td>
<td>161.8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>77.8</td>
<td>77.4</td>
</tr>
<tr>
<td>≃ 15 000</td>
<td>5.53</td>
<td>2.54</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>4.74</td>
<td>1.35</td>
</tr>
<tr>
<td></td>
<td>157.2</td>
<td>72.1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>147.0</td>
<td>36.1</td>
</tr>
</tbody>
</table>

Table 5: Standard deviation (first line \(\times 10^3\)) and maximum (second line \(\times 10^3\)) of the squared errors on the test set for each method and various sizes of the training dataset for N leaching prediction. For each size, the minimal standard deviation and the minimal value of the maxima are in bold. ✓ corresponds to cases impossible to train, either because the model is over-specified (more parameters to estimate than the number of observations: LM2) or because the training size is too large for the method to be used (Dace/SDR/Acosso)
Figure 2: $R^2$ evolution in function of the size of the train set (log scale) for N$_2$O prediction
Figure 3: $R^2$ evolution in function of the size of the train set (log scale) for N leaching prediction
the denitrification chain, being produced by the reduction of nitrate, but being consumed by N_2O denitrifiers. As a consequence, N_2O fluxes are the result of a fragile equilibrium between those processes which are both highly sensitive on environmental conditions such as pH, oxygen availability, substrate availability (Firestone et al., 1979). Thus, N_2O fluxes are characterized by a very high spatial variability and is much harder to predict than nitrogen leaching (Britz and Leip, 2009; Leip et al., 2011a).

- The best results are obtained for the largest training dataset. Mostly, for all methods, the performance increases with the size of the learning dataset despite some exceptions: sometimes, using a larger dataset makes the training process harder and can slightly deteriorate the performance (e.g., for MLP, large datasets leads to harder local minima problems in the optimization procedure: for this method, the best prediction of N leaching estimates is not obtained from the largest training set).

- In a similar way, the variability of the errors tends to decrease with the size of the training dataset but some methods behave differently (see, e.g., Acosso whose variability strictly increases with the size of the training dataset for N leaching prediction).

- In most cases, the most accurate predictions (according to MSE or \( R^2 \) values) are also the predictions that have the smallest variability either from the standard deviation point of view or from the smallest maximum point of view.

Looking deeper into the methods themselves, the following conclusions can also be derived:

- LM1 gives poor performance because the plain linear model is probably too simple to catch the complexity of the modeled phenomenon.

- LM2 performs very badly for small training datasets since it is over-specified (the number of parameters to be estimated is close to the size of the dataset; \( R^2 \) are negative which means that the model is less accurate than the trivial model predicting any observation by the mean value of the outputs). But for large training datasets, it behaved correctly. Additionally, the number of variables selected during the step
AIC, in function of the training dataset size, is given in Table 6. The number of selected variables for N leaching prediction is higher than the number of selected variables for \( \text{N}_2\text{O} \) prediction but it also tends to be more stable regarding the dataset size. Also note that, in any case, the number of selected variables is high compared to the original number of variables (120): this means that the underlying model under study is certainly not plain linear and this explains why LM1 fails to approximate it accurately.

- Splines and kriging based methods have the best performance for small and medium training datasets (especially for N leaching prediction) but they can not be run for large training datasets (up to 2 000 observations) due to the calculation costs. The Dace and SDR models have the best performance. Additionally, the number of selected variables for ACOSSO and SDR are given in Table 7. The number of components effectively included in the model tend decrease with the training set size, especially for \( \text{N}_2\text{O} \) prediction. Comparing this table with Table 6, the number of components is also quite small, even smaller than the number of original variables for some cases.

- Machine learning methods (MLP, SVM and RF) behave correctly for medium training datasets and obtain the best performance for large training datasets. SVM and RF have the best results with a very good overall accuracy, as, for these methods, \( R^2 \) are greater than 90% and 95%, respectively for \( \text{N}_2\text{O} \) and N leaching predictions.

<table>
<thead>
<tr>
<th>Training dataset size</th>
<th>Number of selected variables (( \text{N}_2\text{O} ) prediction)</th>
<th>Number of selected variables (N leaching prediction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>79</td>
<td>95</td>
</tr>
<tr>
<td>500</td>
<td>74</td>
<td>84</td>
</tr>
<tr>
<td>1 000</td>
<td>75</td>
<td>89</td>
</tr>
<tr>
<td>2 000</td>
<td>79</td>
<td>94</td>
</tr>
<tr>
<td>4 000</td>
<td>94</td>
<td>95</td>
</tr>
<tr>
<td>8 000</td>
<td>98</td>
<td>97</td>
</tr>
<tr>
<td>( \approx 15 \ 000 )</td>
<td>96</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 6: Number of variables selected by AIC stepwise procedure in LM2 for \( \text{N}_2\text{O} \) prediction and N leaching prediction in function of the training dataset size
Moreover, Wilcoxon paired tests on the residuals (absolute value) were computed to understand if the differences in accuracy between the best methods were significant: for $N_2O$ prediction, the difference between the best performance (RF) and the second one (SVM) is significant (p-value equal to 0.16%) whereas, for $N$ leaching prediction, the difference between the best performance (SVM) and the second one (RF) is not significant. This test confirms the differences between the best performance of metamodels obtained with different dataset sizes: for example, the difference between SVM trained with about 15 000 observations and Dace trained with 2 000 observations is significant (p-value smaller than $2.2 \cdot 10^{-16}$).

Finally, we took into account the time needed to train the metamodel and subsequently to use it for prediction. The time for training is not so important as it is spent only once during the calibration step. The time for prediction is a key point for CCAT project and so it played a leading role in choosing the best metamodel; it must be quite limited to allow fast multi-scenario simulations or sensitivity analysis. Table 8 provides the approximate time spent to train and use each method with large datasets (respectively, about 15 000 observations for the training step and about 19 000 observations for the prediction one) on a desktop computer.

### 4.4.2. Computational time

The training time for LM1 was the best one but the corresponding performance is very poor. RF had a low training time since it does not require any parameter tuning and it is not very sensitive to the size of dataset thanks to the bootstrapping procedure. The prediction time is really low for all the...
Table 8: Approximative time for training from about 15 000 observations (first line) and for predicting about 19 000 observations (second line) on a desktop computer (Processor 2GHz, 1.5GO RAM). In the case of SDR, ACOSSO, and DACE we report the time for training using samples with 2 000 model runs because the method cannot be used for the largest training datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Train</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM1</td>
<td>&lt;1 s.</td>
<td>&lt;1 s.</td>
</tr>
<tr>
<td>LM2</td>
<td>50 min</td>
<td>90 s.</td>
</tr>
<tr>
<td>Dace</td>
<td>80 min</td>
<td>14 min</td>
</tr>
<tr>
<td>SDR</td>
<td>4 hours</td>
<td>4 min.</td>
</tr>
<tr>
<td>Acosso</td>
<td>65 min</td>
<td>1 s.</td>
</tr>
<tr>
<td>MLP</td>
<td>2.5 hours</td>
<td>1 s.</td>
</tr>
<tr>
<td>SVM</td>
<td>5 hours</td>
<td>20 s.</td>
</tr>
<tr>
<td>RF</td>
<td>15 min</td>
<td>5 s.</td>
</tr>
</tbody>
</table>

Methods compared to the DNDC-EUROPE runs which had demanded about 1 day to simulate the same outputs on a high performance computer cluster. Even though RF was not the fastest approach it provides the best compromise between speed and accuracy. SVM spent more time in prediction since it required the calculation of the kernel matrix whose size is proportional (and thus much more sensitive) to the number of new predictions to make. The same issues applies to splines approach, where the kernel matrix has to be re-computed for every ANOVA term in the decomposition, as well as for kriging, thus explaining the larger computational cost. The highest cost for SDR predictions are linked to the more detailed decomposition, which implies a larger number of reproducing kernels. To compute the large amount of 19 000 model outputs, the time required for predictions does not exceed a few minutes in any cases.

4.4.3. Metamodeling interpretation

To give an indication of which variables are important in the prediction of both inputs, an “importance” measure was calculated for each variable of the best final model (i.e., random forest trained with the full training dataset for \( \text{N}_2\text{O} \) prediction and SVM trained with the full dataset for \( \text{N} \) leaching prediction). For random forests, the importance is quite common: for a given input variable, the values of out-of-sample observations are randomly permuted; the mean squared error is then calculated based on all out-of-sample sets for all trees in the forest. The increase in the mean squared error compared to the out-of-sample mean squared error calculated with the true values of the predictor is called the importance of the predictor (see Genuer et al., 2010 for a complete study of this quantity in the framework of variable selection problems). Unfortunately, MLPs and SVMs are not based on bootstrapping so out-of-sample observations do not exist for these methods. Hence, importance cannot be defined or directly compared to
the one given for random forests. Nervelessness, a close definition can be introduced by using the validation set selected for the tuning process and by comparing the mean squared error of permuted inputs to the true squared error on this validation set.

Figure 4 illustrates the values of the importance measure in both cases. It can be seen that the two metamodels are very different: that (RF) which aims at estimating N\textsubscript{2}O fluxes (left) is mainly based on two important variables (\textit{SOC} and \textit{pH}) whereas SVM, used to estimate N leaching, has a less strict behavior: at least four variables are important in that last modeling, \textit{N\textsubscript{MR}}, \textit{N\textsubscript{FR}}, \textit{pH} and \textit{Nres}.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure4.png}
\caption{Importance measure for each variable in the case of (left) N\textsubscript{2}O prediction with the full training dataset and random forest and of (right) N leaching prediction with the full training dataset and SVM (For the meaning of the acronyms, please refer to Section 4.2).}
\end{figure}

N\textsubscript{2}O fluxes are mainly related to denitrification processes, which require anaerobic conditions and organic material as substrate (Firestone et al., 1979). Anaerobic conditions form if diffusion of oxygen is blocked in wet soils, or in denitrification “hotspots” around organic matter promoting very high oxygen consumption rates (Parkin, 1987). It is therefore not surprising that the soil organic carbon content (\textit{SOC}) was found to be the most important for the prediction of N\textsubscript{2}O fluxes. Soil pH is also an important parameter, influencing both the reduction of nitrate (total denitrification) but also the reduction of N\textsubscript{2}O to N\textsubscript{2} (Granli and Bøckman, 1994). For nitrogen leaching,
on the other hand, we found that the most important factor was the most important factor was nitrogen input as manure amendment, mineral fertilizer spreading, and N from crop residue incorporation in the soil before sowing (these are indeed even more important than pH). To a large degree, nitrogen leaching is determined by soil texture which controls the percolation rate of water through the soil profile and precipitation. As a consequence, it is not surprising to find the top-factors determining nitrogen leaching in a relatively narrow range, as compared to N$_2$O fluxes.

4.5. Conclusion about the comparison of metamodeling strategies

The experiments described in the following subsections enlighten several facts: first, metamodeling strategies were able to approximate accurately N$_2$O and N leaching predictions at a low computational cost. Even with small dataset sizes (100 HSMUs to train the data), the overall accuracy rate, measured by $R^2$, is greater than 80% for at least one metamodel. In this case study, N$_2$O was harder to predict than N leaching. Then, increasing the size of the training dataset is time consuming but also leads to a better accuracy in the prediction for (almost) all the methods. Hence, the selection of a metamodeling approach has to be based on a careful compromise between computational costs and accuracy. This choice strictly depends on the size of available training data and on the project’s target. We pointed out that splines and kriging based methods should be chosen when the number of training data is smaller than 2 000 since they provided the most accurate solution with a reasonable running time. With large datasets, random forests were able to handle the training step and to calculate accurate predictions with low computational costs (more than 15 000 observations were trained in about 15 minutes and only several seconds were needed in predicting 19 000 new values).

Finally, we pointed out, in Section 4.4.3, that combining metamodeling with an importance measure can also be used to provide a simplified insight on the important processes and on the main input variables involved in the prediction of N$_2$O fluxes and N leaching. This can help to find strategies to control nitrogen surplus or to perform a fast sensitivity analysis. This last issue is currently under investigation.
5. Conclusion

This article provides a full comparison of several metamodel approaches for the prediction of N$_2$O fluxes and N leaching from European farmlands. The conclusions of the meta-model comparison are general enough to be extended to other similar case studies. A more valuable and detailed impact assessment of CC standards at European or country level is possible only by simulating all the 207000 HSMUs that cover the EU27. This approach demands the collection of enormous amounts of data and their storage into large datasets. From our work, random forest proved to be a reliable and effective tool for elaborating large datasets with low computational costs and an acceptable accuracy. For these reasons it has been chosen to be integrated into the CCAT platform to estimate the N$_2$O fluxes and N leaching from the EU27 farmlands.

References


first prototype of the analytical tool. Technical Report Deliverable 2.3 of EU STREP 44423-CCAT.


van der Velde, M., Bouraoui, F., de Vries, W., 2009. Derivation of EPIC meta-models to evaluate the impact of cross compliance measures on leaching and runoff of nitrogen and soil erosion at European scale. CCAT Deliverable (report) 4.2.3.2.. CCAT project.


