Supermodeling dynamics and learning mechanisms

Wim Wiegerinck [*1], Miroslav Mirchev [†2], Willem Burgers [‡1] and Frank Selten [§3]

[1]Radboud University, Nijmegen, The Netherlands
[2]Politecnico di Torino, Turin, Italy
[3]Royal Netherlands Meteorological Institute, De Bilt, The Netherlands

To appear in: Consensus and Synchronization in Complex Networks, Kocarev, Ljupco (Ed.), Springer, 2013

Abstract

At a dozen or so institutes around the world, comprehensive climate models are being developed and improved. Each model provides reasonable simulations of the observed climate, each with its own strengths and weaknesses. In the current multi-model ensemble approach model simulations are combined a posteriori. Recently, it has been proposed to dynamically combine the models and so construct one supermodel. The supermodel parameters are learned from historical observations. Supermodeling has been successfully developed and tested on small chaotic dynamical systems, like the Lorenz 63 system. In this chapter we review and discuss several supermodeling dynamics and learning mechanisms. Methods are illustrated by applications to low dimensional chaotic systems: the three dimensional Lorenz 63 and Lorenz 84 models, as well as a 30 dimensional two-layer atmospheric model.

1 Introduction

Machine learning has developed methods and algorithms for automatic modeling by general approximators, such as neural networks. The approximators are optimized on the basis of observational data [3]. This

[*]w.wiegerinck@science.ru.nl
[†]miroslav.mirchev@polito.it
[‡]w.burgers@science.ru.nl
[§]selten@knmi.nl

procedure is called learning or training. Modeling by learning is an efficient alternative for the conventional approach in which models are to be explicitly designed and parameterized in detail by humans.

A standard learning paradigm in machine learning is to fit the parameters of a model by minimizing a cost function on a so-called training set. The choice of the cost function depends on the model task. In conventional time series modeling, the task of the model is to make a prediction of the value at the next time step (the output) given the value at the current time (the input). The training set consists of measurements at subsequent times, which can be viewed as set of (input, target output) examples. For a given parameter setting, one can compute the model output for each of the inputs in the training set. The standard choice for the cost function to be minimized is the sum of squared differences between model and target outputs. To optimize a model for predictions over longer time, multi-step ahead learning has been proposed [13]. The cost functions for these tasks are constructed by selecting a number of initial states from the measured time series and then taking the sum of squared prediction errors, i.e. the errors of the multi-step prediction sequences starting at the selected initial states.

In the context of chaotic systems, often one is not so much interested in the prediction accuracy but more in the long term dynamical behavior of the system. One example is climate modeling. Assuming that the weather system is deterministic chaotic, the climate can be defined as the statistics of the attractor of the weather system [7]. Weather prediction is a prediction of the systems state at a near future time given the observed state at the current time. Climate prediction is to predict statistical properties of the attractor, rather than day-to-day (weather) predictions. Unfortunately, no dedicated algorithm exists to perform attractor learning. Instead, models are trained to make good predictions, and then the model attractor is obtained as a by-product by simulating the model autonomously over a long time [22, 21, 1]. It has been observed that multi-step ahead learning provides better solutions in attractor terms than the more conventional one-step ahead approach. A one-step ahead trained model may yield good short term predictions, starting exactly on a given data point. However, if the model runs autonomously for longer time, due to inevitable model error, the model state drifts away from the observed data due to sensitive dependence on initial conditions and the transition to its own attractor. A multi-step ahead trained model has been tuned to stay close to the data trajectory for a certain amount of time. Another way to improve robustness of learning is to reduce the drift due to sensitive dependence on initial conditions by loosely connecting the model to the data during learning. This will tend to keep the model state close to the data, allowing the models to synchronize with the data and allowing extension of the learning horizon [1, 6].

A problem of applying machine learning methods to climate modeling is the huge dimensionality of the system. A pure data driven machine learning approach starting from scratch is limited to low (order 1 to 10) dimensional systems, since the amount of data needed to tune the model parameters typically scales exponentially with the dimension of the system. This is known as the curse of dimensionality [3]. Models to describe

climate systems have at least several thousands dimensions or millions degrees of freedom in case of the state-of-the-art climate models, which makes a neural network (or similar machine learning method) approach infeasible.

The existing approach to simulate and predict the behavior of real, complex systems like the Earths climate system is to solve the governing equations in a suitable approximate form numerically using discretization techniques. In addition, empirical formulations are implemented for unresolved physical processes. These numerical models contain numerous uncertain parameters that are given values by trial and error ("tuned") in order to produce model simulations that are as close as possible to observations of the real system. A dozen or so very advanced climate models have been developed that differ in discretization techniques, spatial resolution, the range of processes that are explicitly modeled and the empirical formulations of the unresolved processes. Progress over time is achieved by improving the resolution and empirical formulations of the individual models. However, despite the improvements in the quality of the model simulations, the models are still far from perfect. For instance a temperature bias of several degrees in annual mean temperatures in large regions of the globe is not uncommon in the simulations of the present climate [24]. Nevertheless these models are used to simulate the response of the climate system to future emission scenarios of greenhouse gases. It turns out that the models differ substantially in their simulation of the response: the global mean temperature rise varies by as much as a factor of 2 and on regional scales the response can be reversed, e.g. decreased precipitation instead of an increase. It is not clear how to combine these outcomes to obtain the most realistic response. The standard approach is to take some form of a weighted average of the individual outcomes [26]. In [17] an on-line learning approach has been proposed to update weights in the light of past performance of the models.

It has been observed by climate researchers that the imperfections of these models are often complementary; for instance one atmospheric model might produce more realistic heat fluxes for the ocean, while another produces more realistic momentum fluxes. An improved simulation was obtained in a recent study [10] in which two atmospheric models were coupled to one oceanographic model. Importantly, it was not known a priori that one model produced a more realistic heat flux at the ocean surface and that the other model produced a more realistic momentum flux. Rather, those relative advantages became apparent only in the context of the behavior of the entire model with exchanged variables. Running models in parallel and allowing a dynamic exchange of information appears a feasible avenue to increase the simulation performance by combining the strength of the individual models.

In [5, 4], this approach was set in the context of dynamical systems and synchronization theory. The idea is that there is a collection of reasonably accurate models that all aim to simulate the dynamics of a ground truth system. Due to their differences, different models will produce different attractors. By introducing couplings, the states of the different models may fall into a synchronized motion [19, 18]. This could be interpreted as a consensus state. To make the consensus better than the average

of a conventional ensemble without couplings, [5, 4] proposed to make the couplings adaptive, i.e., learn them from data using methods from dynamical systems parameter estimation theory [6] in order to minimize the synchronization error with the observed ground truth system.

A more conventional machine learning approach was proposed in [2]. They interpreted the connections in the interactive ensemble as the parameters of a so-called supermodel. These parameters can be optimized using a cost function. This is basically the same cost function as used in neural networks attractor learning discussed earlier [22, 21, 1]. In [27] it has been noted that the connections of learned supermodels are typically very large, implying that the supermodel will rapidly equilibrate to an (almost) synchronized state. The dynamics of the synchronized state is then approximately described by an averaged dynamics [11, 25]. This suggested an alternative, simplified supermodel approach that takes the averaged dynamics as a starting point [27]. This has the advantage of making the supermodel dynamics linear in its parameters, allowing faster and more scalable learning schemes. This modeling and optimization approach has strong links with ensemble methods in machine learning [20, 12].

Supermodeling with adaptive parameters has only been applied to small scale toy problems. The hope of supermodeling is being able to apply machine learning methods to nonlinear dynamical systems with thousands of variables. This hope is based on the fact that supermodeling starts from existing models that were developed by domain experts and are heavily based on domain knowledge in physics, fluid dynamics and atmospheric sciences, while conventional machine learning using general approximators as basis functions starts from scratch. This should give supermodels a lead in the learning task. Another way to put this is that in supermodeling the internal degrees of freedom are constrained by the underlying models, making the curse of dimensionality much less severe.

In the remainder of this chapter we will review in section 2 supermodel dynamics and learning mechanisms as recently been proposed [4, 2, 27]. We provide in section 3 a set of numerical examples on low dimensional systems that are related to atmospheric science, namely the famous three dimensional Lorenz 63 model [14], the three dimensional Lorenz 84 model[15] and a thirty dimensional truncated two-layer atmospheric model, the T5 quasi-geostrophic baroclinic model [9]. In section 4 we address the issue of robustness of the learning methods against finite information exchange between the imperfect models and against noise in the observations. Then we end with a discussion in section 5.

2 Supermodeling dynamics and learning mechanisms

The general concept of supermodeling could be described as follows. We assume that there is a ground truth dynamical system that we want to model. The dynamical equations of the ground truth cannot be accessed directly, but we assume to have a dataset of observations: a time series of finite length that is generated from the ground truth. In addition, we

have access to a set of M models, labeled by μ. These imperfect models are assumed to provide good but imperfect approximations the ground truth dynamics. The models are defined in terms of their dynamics, given as a system of nonlinear ordinary differential equations (ODEs),

$$\dot{\vec{x}}_\mu = \vec{f}_\mu(\vec{x}_\mu) \tag{1}$$

in which the dot denotes the time derivative. $\vec{x}_\mu$ denotes the D_μ dimensional state vector of model μ. A supermodel is a model that defines a dynamics based on some parametrized combination of the imperfect models

$$\dot{\vec{x}}_{\text{sumo}} = \vec{F}_{\text{sumo}}\left(\vec{x}_{\text{sumo}}; \vec{W}_{\text{sumo}}, \left\{\vec{f}_\mu\right\}\right) \tag{2}$$

in which $\vec{W}_{\text{sumo}}$ are the supermodel parameters. $\vec{x}_{sumo}$ denotes the state of supermodel. The supermodel defines a dynamics in a super state space, which may be larger than the model state space. Additional operations might be required to map model states to supermodel states and back. Typically, the supermodel state space is the direct product of the model state spaces and the back-projection is done by averaging.

The supermodel parameters $\vec{W}_{\text{sumo}}$ can be set manually, or they can be tuned to the data set. Many different approaches can be considered here. Basically these depend on supermodel class, an optimization procedure and a possibly a cost function and a validation criterion. Validation can be done by studying the supermodel behavior, e.g. by comparing with a held-out validation set of the data. In the context of climate modeling, the supermodel would be primarily on the supermodels climatology, (rather than on its short-term weather prediction performance). The optimized and validated supermodel is then to be used as a simulation tool and estimate quantities of interest of e.g. the real climate system.

Both good functional forms for 2 and methods to optimize the supermodel parameters are crucial if supermodels are to be applied to real complex systems as they should improve upon the separate model results as well as on their ensemble averages. A strategy of researching supermodeling methods is to do model simulations with an assumed ground truth dynamics and a set of assumed imperfect models. Here it is to be noted that the assumed ground truth dynamics is not to be used directly in the supermodel optimization, only to generate data sets, both for optimization as for assessment afterwards. In this researching strategy, one can vary the complexity of the ground truth and imperfect models, and zoom in or out on any modeling aspect that one is interested in, which involves not only accuracy but also scalability of the supermodel methodology.

This strategy has been followed in the recently proposed supermodels, which have demonstrated their performance on small scale artificial systems. The ground truth is assumed to be described by a D dimensional ground truth state vector $\vec{x}_{\text{gt}} = (x^1_{\text{gt}}, \ldots, x^i_{\text{gt}}, \ldots, x^D_{\text{gt}})$ that is governed by a ground truth system of ODEs,

$$\dot{\vec{x}}_{\text{gt}} = \vec{f}_{\text{gt}}(\vec{x}_{\text{gt}}) \ . \tag{3}$$

The dynamical equations of the ground truth cannot be accessed directly by the supermodel, it is only used to generate data sets and for a posteriori assessment of e.g. supermodel parameters. In the models that are

considered in this chapter, all model state vectors have the same dimension D, and the different state components $x^i_{\rm gt}$ and x^i_μ represent the same quantity. This should be seen as a first step in the supermodel development. Of course, in reality, the dimensionalities of the ground truth and the various imperfect models are different and it will be nontrivial to relate variables in different models to each other.

2.1 Supermodel dynamics

We will consider two particular classes of supermodel dynamics that have been proposed. The first one is obtained by connecting models [4, 2]. The second one is by averaging models [27].

2.1.1 Connected supermodels

The first class of supermodels is defined as a system of linearly connected imperfect models [4, 2]. We refer to this class as connected supermodels. The imperfect models are connected component wise with each other through positive coefficients $C^i_{\mu\nu} \geq 0$. The connection coefficients $\vec{C} \equiv \{C^i_{\mu\nu}\}$ are the supermodel parameters. The state space of the supermodel is the direct product of the model state spaces. With M models, each of dimension D, the state space is MD dimensional. The (i, μ) component of the supermodel obeys

$$\dot{x}^i_\mu = f^i_\mu(\vec{x}_\mu) + \sum_\nu C^i_{\mu\nu}(x^i_\nu - x^i_\mu) \; . \tag{4}$$

The supermodel state is projected back to the D dimensional ground truth space by taking the average over the models

$$\vec{\overline{x}}_{sumo} = \frac{1}{M} \sum_\mu \vec{x}_\mu \; . \tag{5}$$

The idea is that with positive connections, states of the different models are attracted to each other, so that the states will synchronize, which can be interpreted as a consensus. Due to the asymmetry in the connections, there will be different emphasizes on the different model components of the various models in reaching the ensemble consensus. In [16], learning with unconstrained connections has been explored. It is found that allowing negative connections could help the performance of learning. However, negative connections may cause the states of the individual models to be repelling with a force that is increasing linearly with the distance of the states, possibly leading to divergent model states and thus unstable supermodels. So extra care has to be taken to prevent such instabilities. Negative connections are not further explored in this chapter.

2.1.2 Weighted supermodels

The second class of supermodels is defined as the system in which the dynamics is given by a weighted averaged dynamics of the imperfect models [27]. We refer to this class as weighted supermodels. The model components are averaged with weights w^i_μ, which are the parameters of the

supermodel. They satisfy $w_\mu^i \geq 0$ and $\sum_\mu w_\mu^i = 1$. For the i component, the supermodel obeys

$$\dot{x}^i = \sum_\mu w_\mu^i f_\mu^i(\vec{x}) \; . \tag{6}$$

The dimensionality of the supermodel is equal to the dimensionality of the individual models and the assumed ground truth, so no additional projection is needed. Weighted supermodels can be considered as connected supermodels with infinitely strong connections, i.e. connections of the form $\kappa\vec{C}$ with $C_{\mu\nu}^i > 0$ and $\kappa \to \infty$. In this limit it can be shown that all model states are completely synchronized $\vec{x}_\mu = \vec{x}_\nu$, and that synchronized state follows the weighted averaged dynamics (6). For the relation between $\vec{w}$ and $\vec{C}$ see [27] and a chapter elsewhere in this book.

The weighted supermodel can also be motivated directly, since they can (approximately) be viewed as an concatenation of an ensemble of models from machine learning [20, 12, 3]. The idea behind ensemble of models is that the in expectation, the error of the average is smaller than the average of the errors. This means that if the individual vector fields $\vec{f_\mu}$ are random disturbances from the optimal vector field, the averaged vector field $\frac{1}{M}\sum_\mu \vec{f_\mu}$ is in expectation closer to the optimal vector field than the individual vector fields. If there is no other means to determine which of the individual vector fields is prefered, this makes the unweighted supermodel (i.e., weighted with equal weights $w_\mu^i = 1/M$) already favorable.

2.2 Supermodel learning

The supermodel parameters can be set manually, or they can be learned, i.e., optimized on the basis of a data set of observations. The most obvious manual setting is the classical a posteriori ensemble, in which the models are run independently and averaged afterwards. The classical a posteriori ensemble could be considered as a trivially connected supermodel with all $C_{\mu\nu}^i = 0$. A second supermodel with manually set weights is the unweighted supermodel, motivated in the previous paragraph. We refer to both of these supermodels with manually set parameters as baseline supermodels. Supermodeling combined with learning has only added value if it outperforms these baseline supermodels.

In all learning strategies, it is assumed that there is a training set of observations of the ground truth during a time interval of length T_{train} forming a time series $\{\vec{x}_{\text{gt}}(t_0), \vec{x}_{\text{gt}}(t_0+\tau), \ldots, \vec{x}_{\text{gt}}(t_0+T_{\text{train}})\}$. In this section and the following section, we assume that the time steps τ are the same as the integration step sizes in that are used in the model integrations.

2.2.1 Nudging and parameter dynamics

A learning strategy that is explicitly based on synchronization is outlined in [4]. We refer to this method as the nudging method.

There are two ingredients in this method. The first one is about how the data is entered into the system. This is done by a data assimilation method called nudging [28]. During training, additional connections with

strength $C_{\rm gt} > 0$ are defined that couple the model variables with the data from the ground truth,

$$\dot{x}^i_\mu = f^i_\mu(\vec{x}_\mu) + \sum_{\nu \neq \mu} C^i_{\mu\nu}(x^i_\nu - x^i_\mu) + C_{\rm gt}(x^i_{\rm gt}(t) - x^i_\mu) \ . \tag{7}$$

These additional coupling terms are called nudging terms. Nudging forces the supermodel to remain close to the data. Then the second ingredient is to define, during training, a dynamics to adapt the connections

$$\dot{C}^i_{\mu\nu} = a(x^i_\nu - x^i_\mu)\left(x^i_{\rm gt}(t) - \frac{1}{M}\sum_\mu x^i_\mu\right) - \epsilon/(C^i_{\mu\nu} - C_{\rm max})^2 + \epsilon/(C^i_{\mu\nu} + \delta)^2 \ , \tag{8}$$

where the adaptation rate a is a constant. The terms with coefficient ϵ dynamically constrain all connections $C^i_{\mu\nu}$ to remain in the range $(-\delta, C_{\rm max})$. A Lyapunov function argument can be formulated that shows that the method converges [6].

To produce the simulation results presented later in this chapter, the system with equations (7) and (8) is swept repeatedly through the training set, each time starting with all the $\vec{x}_\mu = \vec{x}_{\rm gt}(t_0)$, but with the $C^i_{\mu\nu}$'s equal to the $C^i_{\mu\nu}$'s of the previous run. After training, $C_{\rm gt}$ is set to zero and the $C^i_{\mu\nu}$'s are kept fixed. Following [4], both ϵ and δ are taken to be 0.01 in our simulations, whereas $C_{\rm max} = 100$ and $C_{\rm gt} = 10$ and $a = 1$.

2.2.2 Cost function optimization

The optimization method described in this paragraph is applicable to both the connected and the weighted supermodels. It relies on the definition of a cost function which is to be minimized with respect to the supermodel parameters. We refer to this method as cost function optimization. The cost function that we consider here is described in [2] and is similar to the one introduced in neural network training [22, 21, 1]. The cost function is constructed by taking initial conditions $\vec{x}_{\rm gt}(t_k)$ at K times t_k from the training time series, $k = 1, \ldots, K$, separated by fixed distances d. Then, starting from these K initializations, short integrations of length Δ are performed with the supermodel (see figure 1). As a measure of the ability of the supermodel to follow the truth for longer periods, the cost function is defined as the mean squared error between the K short trajectories of the supermodel and the ground truth according to the training time series,

$$E(\vec{W}_{\rm sumo}) = \frac{1}{K\Delta}\sum_{k=1}^{K}\int_{t_k}^{t_k+\Delta} ||\vec{\vec{x}}_{sumo}(\vec{W}_{\rm sumo}, t) - \vec{x}_{\rm gt}(t)||^2 \gamma^t dt \ . \tag{9}$$

With the normalization factor $\frac{1}{K\Delta}$, the cost function represents the time averaged mean squared error. Trajectories diverge not only due to model imperfections, but also due to internal error growth: even a perfect model deviates from the truth if started from slightly different initial conditions and leads to a non-zero cost function due to chaos. This implies that

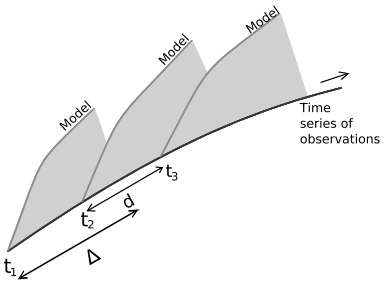

Figure 1: The cost function is based on short integrations of the supermodel starting from observed initial conditions of the truth at times t_k and measures the mean-squared difference between the short evolutions of the supermodel and the truth as indicated by the shaded areas. The short integrations span a time interval Δ and d denotes the fixed time interval between the initial conditions t_k.

the cost function measures a mixture of model errors and internal error growth. Model errors dominate the initial divergence between model and truth, but at later times in the short term integrations internal error growth dominates. These two effects cause a trade off in the choice of the length of the short integrations Δ as well as the decay term γ. Furthermore, there is a trade off in d. The shorter the intervals between the starting points, the more data. But this requires also longer integration times during optimization.

The cost function is a function of the supermodel parameters, as these parameters determine the trajectories followed by the supermodel. In principle any optimization method can be used to minimize the cost function. In this chapter, we use a least mean squares method, but other methods, e.g. evolutionary algorithms could be applied as well [16]. It has been observed that in particular the cost function optimization with connected supermodels is susceptible to slow convergence [2]. To overcome this, an annealing approach has been proposed. The cost function is iteratively minimized for an increasing number of initializations. After each of these minimizations, the solution is taken as a starting point for the next iteration step, in which the cost function is defined with an additional initialization point.

2.2.3 Quadratic programming

The optimization method described in this subsection is applicable to the weighted supermodels only, because it depends on the linearity in the supermodel parameters. This approach views the weighted supermodel as an concatenation of an ensemble of models, so ensemble optimization techniques can be straightforwardly applied. A well-known approach makes use of quadratic programming [20, 12]. To apply this approach to supermodeling, the training time series is considered as a set of (input, output) pairs of a discrete time mapping $\vec{x}(t) \rightarrow \vec{x}(t+\tau)$. Furthermore, We as-

sume small step size $\tau = \Delta t$ and assume that for model integrations of this time length, $\vec{x}(t + \Delta t) = \vec{x}(t) + \dot{\vec{x}}(t)\Delta t$ holds and terms of order Δt^2 can be ignored.

So the training set is the set of pairs $\{(\vec{x}_{\rm gt}(t), \vec{x}_{\rm gt}(t + \Delta t))\}_t$. The i-th component of the error (= mismatch between output and desired output) of model μ on training pair at time t is

$$\epsilon^i_\mu(t) = f^i_\mu(\vec{x}_{\rm gt}(t))\Delta t - (x^i_{\rm gt}(t + \Delta t) - x^i_{\rm gt}(t)) \; . \tag{10}$$

The error of the supermodel for this training pair then follows directly from the errors of the imperfect models,

$$\epsilon^i(t) = \sum_\mu w^i_\mu f^i_\mu(\vec{x}_{\rm gt}(t))\Delta t - (x^i_{\rm gt}(t + \Delta t) - x^i_{\rm gt}(t)) = \sum_\mu w^i_\mu \epsilon^i_\mu(t) \; . \tag{11}$$

Each of the error components $\epsilon^i(t)$ is a function of $\vec{w}^i \equiv (w^i_1, \ldots, w^i_\mu, \ldots, w^i_M)$ only. The sum-squared-error of the supermodel is the sum of sum-squared-errors per component, which can be expressed as

$$\begin{aligned} E^i(\vec{w}^i) &= \sum_t (\epsilon^i(t))^2 \\ &= \sum_t \big(\sum_\mu w^i_\mu \epsilon^i_\mu(t) \big)^2 \\ &= \sum_{\mu\nu} w^i_\mu w^i_\nu \big(\sum_t \epsilon^i_\mu(t)\epsilon^i_\nu(t) \big) \; . \end{aligned} \tag{12}$$

Upon definition of the error correlation

$$\chi^i_{\mu\nu} \equiv \sum_t \epsilon^i_\mu(t)\epsilon^i_\nu(t) \; , \tag{13}$$

the sum-squared-error of the supermodel can be written as

$$E^i(\vec{w}^i) = \sum_{\mu\nu} w^i_\mu w^i_\nu \chi^i_{\mu\nu} \; . \tag{14}$$

To minimize the sum-squared-error, we minimize each of the $E^i(\vec{w}^i)$ separately under the constraints $w^i_\mu \geq 0$ and $\sum_\mu w^i_\mu = 1$. This can be done by quadratic programming. Note that for this optimization, the error per model has to be computed only once to construct the matrices χ^i. This is an important advantage if supermodeling has to be applied to complex systems. Furthermore, the matrices χ^i are postive semi-definite, which makes the functions E^i convex, so there will be no local minima.

2.3 Relation between the methods

The different optimization methods have some relations. For example, the coupling to the ground truth in the nudging method could be seen as a soft and smoothed version of the repeated initializations in the cost function approach. In the nudging approach, instead of hard initializations, the parameterized system remains close to the ground truth during learning due to the couplings with the data. A related difference is that

in the cost function approach the supermodel and ground truth are completely decoupled after initialization, i.e., the supermodel runs free from the ground truth. The training algorithm can thus monitor the error that develops, and tune the parameters to reduce this error. On the other hand in the nudging method there is no fully free model run, since the model remains coupled to the data. This seems to create more a steady state error, which is then monitored and minimized by the training algorithm. Another difference is that the nudging method introduces a dynamics for the super model coupling parameters. This can be seen as a continuous time sequential gradient descent with a quadratic cost function. In standard sequential gradient descent, a cost function $E(\vec{W})$ is written as a sum of costs over the data points, $E(\vec{W}) = \sum_n E_n(\vec{W})$. Data is presented sequentially and at each presentation of a data point, the parameters are updated. When data point n is presented, the parameter vector $\vec{W}^{(t)}$ is updated according to $\vec{W}^{(t+\Delta t)} = \vec{W}^{(t)} - a\vec{\nabla}E_n\Delta t$ [3], where we increment time with Δt. With the cost function approach, the error criterion and optimization procedure are separated. For example, sequential gradient descent could be employed in the cost function approach as well, as well as any other method from numerical optimization.

In the quadratic programming approach, the error criterion is a one-step-ahead error unlike the cost function which is a multi-step-ahead error. This could make the quadratic programming approach less robust to e.g. noise. It could be remarked that a one-step ahead error is more appropriate for the weighted average supermodels than for the connected super models. In connected supermodels, the models need a finite time to synchronize, whereas weighted supermodels are "synchronized" from the start. As an other way to see this, consider a dynamical model $\vec{f}$ with parameters $\vec{W}$. For small Δt, the one step ahead error quadratic cost contribution at time t has the form

$$E_t(\vec{W}) = ||\vec{f}(\vec{x}_{\rm gt}(t);\vec{W})\Delta t - (\vec{x}_{\rm gt}(t+\Delta t) - \vec{x}_{\rm gt}(t))||^2 \ . \qquad (15)$$

Now if we compute this error for a connected supermodel with parameters $C^i_{\mu\nu}$ in which all models are initialized at the ground truth, $\vec{x}_\mu(t) = \vec{x}_{\rm gt}(t)$, we see that the error is independent of the parameters, since all terms $C^i_{\mu\nu}(x^i_\nu(t) - x^i_\mu(t))$ vanish.

3 Numerical examples and results

To illustrate the different approaches of supermodeling numerically, we apply supermodeling in the context of three different models. Two of them, the Lorenz 63 and the Lorenz 84 model, are three dimensional systems. The third one is the T5 quasi geostrophic model (T5) which has 30 dimensions. Each of these models has a set of parameters, that influence in particular the long term behavior of the system. The set up of the experiments follows the one proposed in [2], i.e. one ground truth with standard parameter values and a small number of assumed imperfect models with perturbed parameters. From the ground truth a training time series of relatively short time duration $T_{\rm train}$ is generated. This training set is the only information that the supermodel and its optimization

method may use for optimization. Then there is a test trajectory, which is generated by running the ground truth for longer time duration T_{test}. The methods are assessed by comparing the test trajectory with a supermodel trajectory of comparable duration. The supermodel trajectory is generated by initializing the supermodel from a random state on the ground truth test trajectory. Subsequently, the supermodel is integrated over a longer period. An initial part of the supermodel trajectory is discarded since such initial part might be transient from the ground truth attractor to the supermodel attractor. In other words, the initial part is discarded to remove spurious correlations of the supermodel trajectory with the initial ground truth state. Assessment is done visually by comparing plots of the ground truth and supermodel trajectories. In the Lorenz 63 simulations, we also provide some statistical measures such as the mean and covariance of these obtained data. However, in our opinion, these are in these low dimensional system less informative than the plots.

All integrations are performed using a standard fourth order Runge-Kutta integration method with constant step size. We took the same step size for both ground truth as for the imperfect models and the supermodels. The step size Δt differs, however, per experiment. In the three dimensional models, we took $\Delta t = 0.01$. In the T5 model, we took $\Delta t = 0.1$. Simulations have been performed in Matlab. The cost function minimization is performed using `lsqnonlin` nonlinear least-squares method, and quadratic programming is performed using `quadprog`, both from Matlab's optimization toolbox.

3.1 Lorenz 63

The first model in which the supermodel concept is illustrated is the famous Lorenz 63 equations [14]. The equations for the Lorenz 63 model are

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z \ . \end{aligned} \tag{16}$$

This model is used as a metaphor for the atmosphere, because of its regime changes and unstable nature. We follow the set-up from [2], i.e. one ground truth with standard parameter values that lead to the famous butterfly shaped attractor, and three imperfect models in the same model class but with perturbed parameters, see table 1.

With these perturbations the imperfect models behave quite differently from the truth as can be seen in figure 2. Both model 1 and 2 are attracted to a point, whereas model 3 has a chaotic attractor that has a similar shape a the attractor of the ground truth, but its position is displaced. All models were initiated from the same state taken from the attractor of the ground truth. The transient evolutions towards the attractor are plotted as well. Note that model 1 and model 2 have actually (at least) two fixed points. Which of the fixed point is reached depends on the initial condition. A discussion on how the existence of fixed points and chaotic attractor depends on the parameters is found in e.g., [8].

	σ	ρ	β
Truth	10	28	$\frac{8}{3}$
Model 1	13.25	19	3.5
Model 2	7	18	3.7
Model 3	6.5	38	1.7

Table 1: Lorenz 63: parameters of the assumed ground truth and the three assumed imperfect models.

Following [16], who observed that the required training set sizes in supermodeling was significantly shorter than the ones assumed in [2], we generated a training set simulating the ground truth system for $T_{\text{train}} = 2$ time units. The resulting training set is plotted figure 3. We applied the four different supermodeling approaches that were discussed in the earlier sections. For nudging we took learning parameters as in section 2.2.1. The nudging method swept 10 times through the training set. In the cost function approach, we took cost function parameters $K = 5$, $\Delta = 1$ and $d = 0.1$, so effectively only a training set of duration $T = 1.5$ has been used. The quadratic programming method used all the data.

From looking at the graphs in figure 4, and comparing the metrics in table 2 we see that all learning supermodel approaches are successful in learning a supermodel with an attractor that is close to the ground truth. The results after learning are better than the individual model results, as well as the base-line supermodels. Regarding the base-line supermodels, note that in the uncoupled (a posteriori ensemble) system, the small butterfly in figure 4 is due to the averaging of two fixed points and one normal-size butterfly. This explains the shrinkage of about one third. The attractor of the unweighted supermodel seems to have already a better shape than the individual models. This suggests that straightforward unweighted averaging may already be a helpful method for model enhancement. The attractor of the unweighted supermodel is generated by simulations starting from the attractor of the ground truth. We remark that starting from other initial states, far from the ground truth attrac-

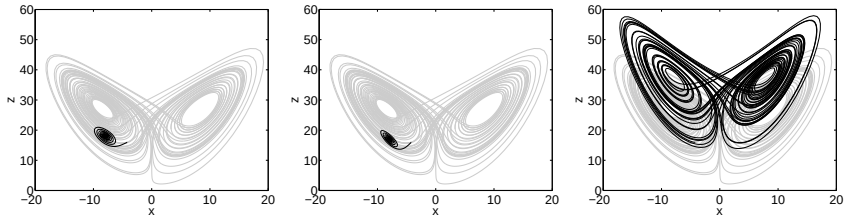

Figure 2: Lorenz 63. Results of imperfect model integration projected on (x, z) plane over a time period of $T = 50$ time units, starting on the starting on the ground truth attractor. From left to right: model 1 to model 3. Gray: assumed ground truth. Black: assumed imperfect models.

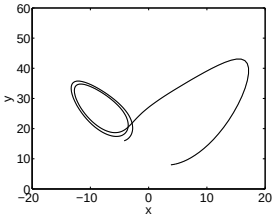

Figure 3: Lorenz 63. Training set of $T_{\text{train}} = 2$ time units generated by the assumed ground truth.

	mean x	mean y	mean z	SD x	SD y	SD z	cov xy	cov xz	cov yz
Truth	-0.3(1.2)	-0.3(1.2)	23.6(0.2)	7.8(0.1)	8.9(0.1)	8.6(0.2)	61.3(1.6)	-0.9(4.1)	-0.8(3.3)
Model 1	-7.9(0.0)	-7.9(0.0)	18.0(0.0)	0.0(0.1)	0.0(0.1)	0.0(0.1)	0.0(0.0)	-0.0(0.0)	-0.0(0.0)
Model 2	-7.9(0.0)	-7.9(0.0)	17.0(0.0)	0.0(0.1)	0.0(0.1)	0.0(0.1)	0.0(0.0)	-0.0(0.0)	0.0(0.0)
Model 3	0.2(0.8)	0.2(0.8)	34.3(0.1)	7.6(0.0)	9.4(0.1)	8.7(0.1)	57.7(0.7)	0.4(2.2)	0.5(3.3)
Uncoupled	5.3(0.3)	5.3(0.3)	23.1(0.1)	2.5(0.1)	3.1(0.1)	2.9(0.1)	6.5(0.4)	0.0(0.3)	0.0(0.4)
Uniform W	0.6(0.8)	0.6(0.8)	20.0(0.3)	7.6(0.1)	8.8(0.1)	8.6(0.2)	58.2(1.7)	2.5(3.6)	1.4(1.7)
Nudging C	0.1(1.1)	0.1(1.1)	23.3(0.1)	7.7(0.1)	8.9(0.1)	8.6(0.0)	58.2(1.3)	0.3(3.8)	0.2(3.5)
Costfn C	-0.2(0.6)	-0.2(0.6)	23.2(0.1)	7.8(0.0)	8.9(0.0)	8.8(0.0)	59.6(0.6)	-0.5(2.0)	-0.5(1.9)
Costfn W	-0.9(1.1)	-0.9(1.1)	23.6(0.1)	7.8(0.2)	8.9(0.2)	8.6(0.1)	60.9(2.4)	-2.9(3.7)	-2.1(3.1)
Quadprog W	-0.5(1.0)	-0.5(1.0)	23.6(0.1)	7.9(0.1)	9.0(0.1)	8.6(0.1)	61.8(1.6)	-1.5(3.1)	-1.2(2.6)

Table 2: Lorenz 63. Means, standard deviations, and covariances (cov) for the ground truth, the imperfect models and the supermodels. Results are based on 10 consecutive runs of $T = 50$ time units, the first run starting from the ground truth attractor. Between brackets: standard deviation.

tor, the unweighted supermodel converges to a point attractor. in other words, the unweighted supermodel contains multiple attractors. This can be understood from the averaged parameters, which are $\sigma = 8.92$, $\rho = 25$, $\beta = 2.97$. For these values, it is known that chaos and fixed points may coexist [8].

Regarding the computational cost of the different optimization methods we found considerable differences in the required CPU time, see table 3. However, it should be remarked that computational costs depends strongly on how the method is exactly implemented. Among others, the convergence criterion is an important parameter. In our simulations, the converge criterion has been not optimized, but fixed in advance as described based on loose criteria on pilot studies. So these CPU time results are only indicative. However, it is save to conclude that in computational costs, the weighted supermodel with quadratic programming outperforms all other methods significantly with about a factor of 100.

3.2 Lorenz 84

The Lorenz 84 system was proposed by Lorenz as a toy model for the general atmospheric circulation at mid latitudes [15]. The model equations

Nudging C	Costfn C	Costfn W	Quadprog W
136	2360	124	1

Table 3: Lorenz 63. relative CPU time costs of the different optimization methods

	a	b	F	G
Truth	0.25	4	8	1
Model 1	0.33	5.2	10.4	0.7
Model 2	0.18	5.2	5.6	1.3
Model 3	0.18	2.7	10.4	1.3

Table 4: Lorenz 84. parameters of the assumed ground truth and the three assumed imperfect models.

are

$$\begin{aligned} \dot{x} &= -y^2 - z^2 - ax + aF \\ \dot{y} &= xy - bxz - y + G \\ \dot{z} &= bxy + xz - z \ . \end{aligned} \tag{17}$$

The x variable represents the intensity of the globe-encircling westerly winds and y and z represent a traveling large-scale wave that interacts with the westerly wind. Parameters F and G are forcing terms representing the average north-south temperature contrast and the east-west asymmetries due to the land-sea distribution respectively. Following [2], we take the parameters of the ground truth and the three imperfect models as in 4. With these parameters the attractor of the imperfect models differ substantially from the truth (see figure 6). The ground truth is chaotic, while all imperfect models have periodic attractors.

The training set was generated by simulating the ground truth for $T_{\text{train}} = 10$, see figure 5. Again, supermodels were optimized with the four different methods, i.e. connected with nudging and cost function, and weighted with cost function and quadratic programming. The parameters of the cost function were $\Delta = 1$, and $d = 1$, $K = 5$, so effectively only the first $T = 6$ time units were used. Figure 7 shows the trajectories of the two base-line and four optimized supermodels. The test trajectories run over a period of $T_{\text{test}} = 100$, of which we plotted the second half period of 50 time units to avoid spurious correlations with the initial state. From looking at the graphs in figure 4, and comparing the metrics in table 2 we see that except nudging, all learning supermodel approaches are successful in learning a supermodel with an attractor that is close to the ground truth. The results after learning are better than the individual model results, as well as the base-line supermodels. The nudging approach seems to be stuck in a local minimum or plateau in its error landscape. This may be due to suboptimal parameter setting. However, this has not been explored further. For the other three optimized supermodels, results after learning looks better than the individual model results, as well as the base-line supermodels. Regarding the computational cost of

Nudging C	Costfn C	Costfn W	Quadprog W
170	501	38	1

Table 5: Lorenz 84. Relative CPU time costs of the different optimization methods

the different optimization methods we found again considerable differences in the required CPU time, see table 5. The same conclusions as in the Lorenz 63 case could be drawn: CPU time results are only indicative, however, it is save to conclude that in computational costs, the weighted supermodel with quadratic programming outperforms all other methods significantly.

3.3 T5 quasi-geostrophic baroclinic model

An important instability mechanism that leads to the growth and decay of the mobile weather systems (also referred to as depressions, low-pressure systems, storms, synoptic eddies, transient eddies) at mid-latitudes is lacking. It is the baroclinic instability mechanism, a process in which the available potential energy stored in the equator to pole temperature gradient is converted into the kinetic energy of the storms. These storms in turn transport heat polewards, thereby reducing the temperature gradient and the cause for their existence. The temperature gradient is continuously restored by the differential heating of the earth by the incoming solar radiation and the outgoing thermal radiation. To model this instability mechanism [9] developed a two-level, hemispheric, quasi-geostrophic spectral model on the sphere, triangularly truncated at wave number five. We refer to this model as the T5 model. A complete description is given in [9] and a summary is given below. The vorticity equation is applied to the 250 hPa and the 750 hPa level, the heat equation is applied to the 500 hPa level. In the following, ψ_1 denotes the 250 hPa streamfunction, ψ_3 the 750 hPa streamfunction, $\psi = \frac{1}{2}(\psi_1 + \psi_3)$ the interpolated 500 hPa streamfunction and $\tau = \frac{1}{2}(\psi_1 - \psi_3)$ the 250-750 hPa thickness. The equations have been nondimensionalized using the earth's radius a as unit of length and the inverse of the angular velocity of the earth as unit of time. A closed set of evolution equations for ψ and τ is obtained by eliminating the vertical velocity:

$$\begin{aligned}
\frac{\partial}{\partial t}\Delta\psi &= -J(\psi,\Delta\psi) - J(\tau,\Delta\tau) - J(\psi,f) + \\
&\quad \frac{f_0}{2}J(\tau-\psi,h) + \frac{C}{2}\Delta(\tau-\psi+\psi_3^*) \\
\frac{\partial}{\partial t}(\Delta - 2\Lambda^2)\tau &= -J(\psi,\Delta\tau) - J(\tau,\Delta\psi) - J(\tau,f) + \\
&\quad \frac{f_0}{2}J(\psi-\tau,h) + \frac{C}{2}\Delta(\psi-\tau-\psi_3^*) + \\
&\quad 2\Lambda^2 J(\psi,\tau) - 2\Lambda^2 Q(\tau^*-\tau) \qquad (18)
\end{aligned}$$

where f_0 the value of the Coriolis parameter at a latitude of 45^0, h the surface topography, C the Ekman damping coefficient, ψ_3^* the lower level streamfunction forcing, τ^* the thickness forcing and Q the cooling coefficient. The Rossby radius of deformation Λ^{-1} is defined by:

$$\Lambda^2 = \frac{f_0^2}{\sigma(\Delta p)^2} \tag{19}$$

where σ is the static stability parameter and Δp the pressure difference between the two levels. The value of Λ^2 is 90 which corresponds to a Rossby radius of deformation of 670 km. The streamfunctions are projected onto a basis of spherical harmonics. The streamfunction is now approximated as follows:

$$\psi(\lambda, \mu, t) = \sum_{n=1}^{5} \sum_{\substack{m=-n \\ m+n=\text{odd}}}^{+n} \psi_{m,n}(t) Y_{m,n}(\lambda, \mu) \ . \tag{20}$$

The restriction to modes with $m + n$ odd excludes currents across the equator. This makes the model hemispheric. The expansion includes three zonal modes and six wave modes. With this choice ψ and τ are characterized by 15 coefficients each so the phase space of the model is 30-dimensional.

The forcings ψ_3^* and τ^* are in the $Y_{0,1}$ mode only. These terms correspond to forcing a westerly zonal wind in the lower level of about 8 m/s and an equator to pole temperature difference of 110 K which produces a zonal wind shear of about 24 m/s between upper and lower level. The Ekman damping coefficient C and the cooling coefficient Q are given a value of 0.01, which corresponds to an e-folding time of around 16 days. The topography h is described with the $Y_{2,3}$ mode and is introduced to destabilize the enforced zonal flow and to locate preferential circulation patterns at fixed geographical positions. The amplitude of h is 0.04 which corresponds to a height of 1.6 km.

By projecting the model equations (18) onto the selected spherical harmonics a system of 30 coupled non-linear ordinary differential equations is obtained describing the evolution of the expansion coefficients. The general form of these equations is given by:

$$\frac{d}{dt}\psi_i = \alpha_i + \sum_{j=1}^{30} \beta_{ij}\psi_j + \sum_{j=1}^{30}\sum_{k=1}^{30} \gamma_{ijk}\psi_j\psi_k \qquad i = 1, .., 30 \ . \tag{21}$$

The constant terms α_i correspond to the forcing terms $\frac{C}{2}\Delta\psi_3^*$ and $2\Lambda^2 Q\tau^*$ thus α_i is non-zero in the evolution equation of the (0,1)-mode only. The linear terms correspond to the cooling and damping terms, the interaction with topography and the Coriolis term. The quadratic terms are the result of the advection of relative vorticity and thickness. Due to the special properties of the spherical harmonics, many of the interaction coefficients γ_{ijk} are equal to zero. Furthermore γ_{ijk} is equal to γ_{ikj}.

The resulting nonlinear ordinary differential equation displays chaotic, regime like behavior with an estimated attractor dimension of around 11.5

	h	τ^*
Truth	0.04	-0.03
Model 1	0.04	-0.04
Model 2	0.06	-0.028

Table 6: T5 model: Parameters of the assumed ground truth and the two assumed imperfect models.

[23]. The most prominent component of the model circulation is an eastwards moving wave. The wave is accelerated and decelerated by the varying intensity of the zonal wind. During short time intervals, it may move westwards. All other waves continuously move westwards. The influence of the zonal wind is insufficient to keep them at fixed geographical longitude. The topography and the barotropic part of the mean state of a 15 year integration is plotted in figure 8. The influence of the topography on the mean state is clearly visible.

To give an impression of the complexity of the T5 model, we plotted a state represented as the stream function at the 750 and 250 hPa level in figure 9. Wavy disturbances are visible in a westward flow. The disturbances grow and decay and travel eastward.

3.3.1 Supermodeling

We took this model as the ground truth and assumed two imperfect models by perturbing h and τ^* with values as in tabel 6. We then combined these in a weighted supermodeling,optimized by quadratic programming. Cost function optimization of both weighted and connected supermodel has been attempted, but failed due to local minima and/or memory problems, depending on the supermodel, the cost function parameter settings and the optimization options. Perhaps fine tuning of the optimization method could remedy this, but at least it indicates that cost function optimization is much more cumbersome than the quadratic programming approach, and it is likely that it will require orders of magnitudes more computational resources. Therefore this has not been pursued further.

Trajectories of both imperfect models as well both base-line supermodels and the optimized weighted supermodel are presented in Fig 10. The chosen phase plane (ψ_{14}, ψ_{15}) displays the components of an unstable wave traveling around the globe. The chaotic growth and decay of this wave in the ground truth is much better simulated in the optimized supermodel as compared to the imperfect models. Unweighted averaging seems to improve a bit on the amplitude as compared to the imperfect models, but unweighted averaging does not seem to capture the dynamics.

4 Limited information exchange, coarse grained training data and noise

The analyses in the previous sections assumed continuous information exchange between individual models composing a supermodel and con-

tinuously available truth data during learning. However, in reality these assumptions may not hold due to computational inefficiency of information exchange and finiteness and imperfections of measurements and the impact of these limitations on performance of supermodels is in the focus of this section.

In order to reduce the inefficiency of information exchange the individual models can be run autonomously and communicate solely at large discrete time steps τ_{exchange}. The supermodel dynamics would then consist of series of successive integrations of the individual models with a step size Δt over a period τ_{exchange} defining mappings

$$x^i_\mu(t + \tau_{\text{exchange}}) - x^i_\mu(t) = F^i_\mu(\vec{x}_\mu(t); \tau_{\text{exchange}}) \tag{22}$$

followed by a certain form of exchange depending on the supermodel class.

In the case of weighted supermodels at the time of exchange all individual models are reset to the weighted average of their states as shown on fig. 4. The supermodel states are also defined as the weighted average of the individual states. Thus, the supermodel dynamics is defined by the successive mappings obtained as a weighted combination of short integrations of the individual models

$$x^i(t + \tau_{\text{exchange}}) - x^i(t) = \sum_\mu w^i_\mu F^i_\mu(\vec{x}(t); \tau_{\text{exchange}}). \tag{23}$$

In connected supermodels the information exchange can be expressed in the form of impulsive coupling between the models, instead of continuous coupling. This form can be seen as a connected supermodel having connection coefficients with values different than zero only at the times of exchange. The supermodel states are again the average of the individual models' states.

A second realistic issue is posed by the limitations of measuring procedures as they usually provide a time series of observations only at time steps τ_{data} larger than integration step sizes Δt. Measurements are used only for training, therefore, we just need to accordingly adapt the learning methods. In the cost function optimization methods this only reduces the number of available error samples and the number of possible states to which individual models can be initialized.

In quadratic optimization when there are data missing so the training sample becomes $(\vec{x}(t), \vec{x}(t + \tau_{\text{data}}))$ and the error components of the imperfect models are

$$\epsilon^i_\mu(t) = F^i_\mu(\vec{x}(t), \tau_{\text{data}}) - (x^i(t + \tau_{\text{data}}) - x^i(t)) \tag{24}$$

By including these components in (14) and minimizing $E^i(\vec{w}^i)$ we can find the weights of the supermodel. We should note that to exchange information between steps of τ_{data}, i.e. when $\tau_{\text{data}} > \tau_{\text{exchange}}$, we need to calculate the weighted average at times τ_{exchange}, which makes the problem non-quadratic. One way to circumvent this problem is to approximate the missing data at steps τ_{exchange}, however, here we restrict the quadratic optimization to $\tau_{\text{exchange}} \geq \tau_{\text{data}}$.

In order to examine the robustness of the supermodels to the limitations described in this section we use the Lorenz 63 system. We assume

	μ_x	μ_y	μ_z	σ_x	σ_y	σ_z	cov_{xy}	cov_{xz}	cov_{yz}
Truth	$0.1_{0.1}$	$0.1_{0.1}$	$23.6_{0.0}$	$7.9_{0.0}$	$9.0_{0.0}$	$8.6_{0.0}$	$62.7_{0.0}$	$0.2_{0.3}$	$0.1_{0.3}$
Costfn C $^{\tau_x=0.01}_{\tau_d=0.01}$	$0.0_{0.1}$	$0.0_{0.1}$	$23.5_{0.0}$	$7.9_{0.0}$	$8.9_{0.0}$	$8.6_{0.0}$	$62.4_{0.0}$	$-0.1_{0.3}$	$-0.1_{0.3}$
Costfn C $^{\tau_x=0.1}_{\tau_d=0.01}$	$0.0_{0.1}$	$0.0_{0.1}$	$22.8_{0.0}$	$7.6_{0.0}$	$8.8_{0.0}$	$8.8_{0.0}$	$56.9_{0.0}$	$0.0_{0.2}$	$0.0_{0.3}$
Costfn C $^{\tau_x=0.01}_{\tau_d=0.1}$	$0.0_{0.1}$	$0.0_{0.1}$	$23.4_{0.0}$	$7.7_{0.0}$	$8.8_{0.0}$	$8.5_{0.0}$	$60.4_{0.0}$	$0.0_{0.3}$	$0.0_{0.2}$
Costfn C $^{\tau_x=0.1}_{\tau_d=0.1}$	$0.0_{0.1}$	$0.0_{0.1}$	$22.5_{0.0}$	$7.3_{0.0}$	$8.6_{0.0}$	$8.5_{0.0}$	$53.4_{0.0}$	$-0.1_{0.2}$	$-0.1_{0.2}$
Costfn W $^{\tau_x=0.01}_{\tau_d=0.01}$	$0.0_{0.1}$	$0.0_{0.1}$	$23.7_{0.0}$	$7.8_{0.0}$	$9.1_{0.0}$	$8.5_{0.0}$	$62.9_{0.0}$	$-0.1_{0.3}$	$-0.1_{0.3}$
Costfn W $^{\tau_x=0.1}_{\tau_d=0.01}$	$0.0_{0.1}$	$0.0_{0.1}$	$23.3_{0.0}$	$8.2_{0.0}$	$9.3_{0.0}$	$8.9_{0.0}$	$67.8_{0.0}$	$0.1_{0.2}$	$0.1_{0.2}$
Costfn W $^{\tau_x=0.01}_{\tau_d=0.1}$	$0.0_{0.1}$	$0.0_{0.1}$	$23.6_{0.0}$	$7.8_{0.0}$	$9.1_{0.0}$	$8.6_{0.0}$	$62.6_{0.0}$	$-0.1_{0.3}$	$-0.1_{0.3}$
Costfn W $^{\tau_x=0.1}_{\tau_d=0.1}$	$0.0_{0.1}$	$0.0_{0.1}$	$23.2_{0.0}$	$7.9_{0.0}$	$9.3_{0.0}$	$8.9_{0.0}$	$65.3_{0.0}$	$0.2_{0.3}$	$0.1_{0.2}$
QP W $^{\tau_x=0.01}_{\tau_d=0.01}$	$0.1_{0.1}$	$0.1_{0.1}$	$23.5_{0.0}$	$7.8_{0.0}$	$9.0_{0.0}$	$8.6_{0.0}$	$62.5_{0.1}$	$0.3_{0.4}$	$0.3_{0.3}$
QP W $^{\tau_x=0.01}_{\tau_d=0.1}$	$-0.1_{0.1}$	$-0.1_{0.1}$	$22.9_{0.0}$	$7.9_{0.0}$	$9.1_{0.0}$	$8.7_{0.0}$	$62.5_{0.0}$	$-0.2_{0.4}$	$-0.2_{0.3}$
QP W $^{\tau_x=0.1}_{\tau_d=0.01}$	$0.1_{0.1}$	$0.1_{0.1}$	$23.4_{0.0}$	$7.6_{0.0}$	$9.1_{0.0}$	$8.9_{0.0}$	$60.5_{0.0}$	$0.3_{0.5}$	$0.2_{0.3}$
QP W $^{\tau_x=0.1}_{\tau_d=0.1}$	$-0.1_{0.1}$	$-0.1_{0.1}$	$23.2_{0.0}$	$7.7_{0.0}$	$9.1_{0.0}$	$9.0_{0.0}$	$60.5_{0.0}$	$-0.2_{0.5}$	$-0.1_{0.3}$

Table 7: Means (μ), standard deviation (σ) and covariance (cov) of supermodels with various steps of information exchange (τ_x) and data (τ_d) obtained by cost function minimization of connected (Costfn C) and weighted (Costfn W) supermodels and quadratic optimization of weighted (QP W) supermodel. The test results are obtained by multiple runs and the subscripts are the corresponding 95% confidence bounds.

we have $T_{\text{train}} = 10$ of training data and $T_{\text{test}} = 1000$ of data used for evaluation of the supermodels. All integrations were done by a Runge-Kutta method of fourth order with step size $\Delta t = 0.01$. The optimizations were performed with the cost function approach and with the quadratic programming approach as described earlier. The obtained results are given in table 7 and it can be seen that although all supermodels remain near the true dynamics their performance only slightly degrades as either τ_{exchange} or τ_{data} rise. The degradation is larger with the increase of τ_{exchange} than with the increase of τ_{data}. Another observed aspect is that weights found by quadratic optimization for one τ_{exchange} in general work fine also for other τ_{exchange} as the optimization results in finding a global optimum. For example, the results in table 7 for ($\tau_{\text{exchange}} = 0.01, \tau_{\text{data}} = 0.1$) were performed using a supermodel trained with ($\tau_{\text{exchange}} = \tau_{\text{data}} = 0.1$). On the other hand, this conclusion in general does not hold for supermodels obtained by cost function optimization.

A final issue that we address is the robustness against noise in the measurements. Therefore, we add a white noise $\mathcal{N}(0, 1)$ to each of the observed variables with overall signal-to-noise ratio (SNR) of 18.4dB. We performed simulations with the different supermodels in the same manner as in the previous analyses and the results are given in table 8. The cost function optimization of weighted supermodels proved to be among all methods most robust to noise along with the other considered limitations. On the other hand, the quadratic programming approach proved to be sensitive to noise and except in the case when $\tau_{\text{data}} = \tau_{\text{exchange}} = 0.01$, failed to find proper weights for all other cases. This is in agreement with earlier observations that a multi-step ahead approach is advantageous

	μ_x	μ_y	μ_z	σ_x	σ_y	σ_z	cov_{xy}	cov_{xz}	cov_{yz}
Truth	$0.1_{0.1}$	$0.1_{0.1}$	$23.6_{0.0}$	$7.9_{0.0}$	$9.0_{0.0}$	$8.6_{0.0}$	$62.7_{0.0}$	$0.2_{0.3}$	$0.1_{0.3}$
Costfn C τ_x=0.01 τ_d=0.01	$0.1_{0.2}$	$0.1_{0.2}$	$22.8_{0.0}$	$7.5_{0.0}$	$8.5_{0.0}$	$8.3_{0.1}$	$57.0_{0.2}$	$0.3_{0.7}$	$0.3_{0.5}$
Costfn C τ_x=0.1 τ_d=0.01	$0.0_{0.1}$	$0.0_{0.1}$	$22.8_{0.0}$	$7.2_{0.0}$	$8.4_{0.0}$	$8.1_{0.0}$	$51.8_{0.0}$	$0.0_{0.3}$	$0.0_{0.3}$
Costfn C τ_x=0.01 τ_d=0.1	$0.1_{0.2}$	$0.1_{0.2}$	$24.8_{0.0}$	$7.7_{0.0}$	$8.2_{0.0}$	$6.0_{0.1}$	$59.2_{0.2}$	$0.3_{0.4}$	$0.2_{0.3}$
Costfn C τ_x=0.1 τ_d=0.1	$-2.1_{2.8}$	$-2.1_{2.9}$	$25.2_{0.0}$	$0.4_{0.1}$	$0.5_{0.1}$	$0.6_{0.1}$	$0.2_{0.1}$	$-0.1_{0.1}$	$0.0_{0.1}$
Costfn W τ_x=0.01 τ_d=0.01	$0.0_{0.1}$	$0.0_{0.1}$	$22.9_{0.0}$	$7.6_{0.0}$	$9.2_{0.0}$	$9.0_{0.0}$	$60.0_{0.1}$	$-0.2_{0.4}$	$-0.1_{0.3}$
Costfn W τ_x=0.1 τ_d=0.01	$0.0_{0.1}$	$0.0_{0.1}$	$22.3_{0.0}$	$7.5_{0.0}$	$8.6_{0.0}$	$9.3_{0.0}$	$54.2_{0.0}$	$0.1_{0.3}$	$0.0_{0.2}$
Costfn W τ_x=0.01 τ_d=0.1	$0.0_{0.1}$	$0.0_{0.1}$	$22.9_{0.0}$	$7.4_{0.0}$	$8.6_{0.0}$	$8.4_{0.0}$	$56.8_{0.0}$	$-0.1_{0.3}$	$0.0_{0.3}$
Costfn W τ_x=0.1 τ_d=0.1	$0.0_{0.1}$	$0.0_{0.1}$	$23.4_{0.0}$	$7.9_{0.0}$	$8.9_{0.0}$	$8.8_{0.0}$	$59.4_{0.0}$	$0.2_{0.4}$	$0.1_{0.2}$
QP W τ_x=0.01 τ_d=0.01	$0.3_{0.7}$	$0.3_{0.7}$	$23.3_{0.3}$	$7.5_{0.6}$	$8.6_{0.7}$	$8.3_{0.6}$	$61.7_{4.9}$	$-0.2_{0.3}$	$-0.1_{0.2}$

Table 8: Means (μ), standard deviation (σ) and covariance (cov) of the different supermodels with various steps of information exchange (τ_x) and data (τ_d) in presence of noise $\mathcal{N}(0,1)$. The test results are obtained by multiple runs and the subscripts are the corresponding 95% confidence bounds.

in learning. The connected supermodels showed variable performance in different optimizations and tests and their performance degrades as τ_{exchange} increases.

5 Discussion

We discussed several variants of supermodeling as a potential tool for climate research as proposed in [5] and further developed in [4, 2, 27]. The goal of supermodeling is to dynamically combine existing good, but imperfect models to a so-called supermodel. Optimization of the supermodel parameters, which control how the dynamical combination is exactly performed, is based on observational data. The hope of supermodeling is being able to improve upon the existing imperfect models in domains with thousands of variables such as needed for climate modeling. For a conventional machine learning approach, we think this is unlikely to be feasible, since such an approach starts in a way from scratch by using general approximators and being purely data driven without hardly using any prior knowledge. Supermodeling, however starts from existing models that were developed by domain experts and are heavily based on domain knowledge in physics, fluid dynamics and atmospheric sciences. This should give supermodels big advantage in the learning task.

Two recently proposed types of supermodels have been discussed. The first one is the connected supermodel, in which imperfect models are connected to each other and influence each other via these connection terms. The other one is the weighted supermodel, in which the supermodel dynamics is a weighted average of the imperfect model dynamics. Simulations suggest that both the connected and the weighted supermodels are able to improve upon the individual imperfect models or the a posteriori ensemble method. Simulations also indicate that supermodeling is robust against limited data availability, limited data exchange and to some

extend to noise in the observations.

Optimization of the supermodels require choices in the parameter settings of the optimization procedure. For instance in the nudging approach, the learning parameter a and the nudging parameters C_{gt} need to be chosen. In the cost function approach, K Δ, and d were set, as well as an annealing scheme. Other important issue is the setting of convergence criteria, which were set rather arbitrary in the simulations in this chapter. All these choices will be of influence on the supermodel solutions and convergence time of the optimization procedure. The optimal setting will depend on the problem at hand, and suboptimal choices may influence the learning result considerable. In the simulations presented in this chapter, we set the optimization parameters rather arbitrary on the basis of a limited number of pilot runs. We did not further try to optimize these settings, but for a realistic application of supermodeling, this is an important issue that needs further investigation.

The connected supermodel proposed in [5, 4, 2] relies on a synchronization between the models. Typically, optimization of connected supermodels leads to large connections. It can be shown that with large connections, dynamics of the synchronized state follows the dynamics according to weighted averages of the imperfect model components. Weighted supermodels can therefore be interpreted as as a "hard coupling" ($\vec{C} \to \infty$) limit of connected supermodels. A disadvantage of a "hard coupled" weighted supermodel could be that the resulting dynamicsl system has less flexibility than with a "soft coupled" connected supermodel. Hard coupling may prevent escapes from individual model states and thus hinder transitions between regimes in the attractor. In another chapter in this book this effect is illustrated in Lorenz 63 system in which one variable is connected. On the other hand, weighted supermodels seem to have many practical advantages. The most important is the existence of a scalable learning schemes such as the quadratic programming method. Its computational advantage is already apparent in the supermodel applications to the three dimensional Lorenz 63 and Lorenz 84 models. In the 30 dimensional quasi-geostrophic baroclinic T5 model, the weighted supermodel with quadratic programming was the only one in which learning converged to an improved solution. Other advantages are interpretability and transparency, the elimination of equivalent solutions, and possibly performance guarantees (see e.g. ensemble methods in [3]).

In real applications, the availability of data as well as the amount of data exchange between models may be limited due to limitations in resources. Therefore an issue is the robustness of the different approaches against these limitations. We tested the robustness of the various approaches in the context the Lorenz 63 model. In general all approaches showed some robustness against the discreteness of information exchange and data availability. Quadratic programming, however, turned out to be less robust against noise than the computationally more intensive cost function approach. This is in a way to be expected, because the quadratic programming approach is basically an optimization method for one step ahead prediction, while the cost function approach is a multi-step ahead method, which is known to be helpful for learning. A hybrid optimization form may be required that combines the advantages of both approaches.

There are still many open problems to be solved before supermodeling can be reliably applied for climate results. An important issue is if supermodeling does not overfit in systems of high dimension. To investigate this, [2] proposed to study supermodeling performances in systems of increasing complexity. One way is to overcome overfitting is to further constrain the connections or weights, or to use regularization methods from machine learning. It would also be interesting to have an quantification of the supermodel uncertainty, such as in the conventional uncoupled ensemble approach where there is a clear prediction mean and variance. A practical issue is how supermodeling is performed with the currently available software in large climate models. Probably a fully connected or fully weighted approach is infeasible, meaning that only subsets of the model variables can be coupled. Another issue is that transformation of variables may be needed prior to the coupling, e.g. due to differences in spatial grid sizes in the models.

A major caveat of using supermodeling for climate prediction is that the super-model is trained on historical data and in a climate prediction problem is subsequently applied to simulate the response of the system to an external forcing, e.g. due to increased greenhouse effect. It is not guaranteed that the supermodel will also simulate this response more realistically, since the response was not part of the training. For the applicability of supermodeling in climate science, it is therefore of crucial importance to get better insight in the physical assumptions that are implied in the supermodeling approach or to devise other methods of model verification, e.g biased to particular atmospheric conditions which mimic the external forcing situations, to obtain confidence in the supermodel predictions. Without these, the super-model approach is more likely to be successful in weather- and seasonal predictions since the cases to be predicted remain closer to the cases present in the training set and data based model verification can be performed easier.

To conclude, whether the supermodeling approach will be beneficial in the context of complex climate models remains to be seen. However, in our opinion, supermodeling is a promising approach that is worthwhile to be explored further.

acknowledgement This work has been supported by FP7 FET Open Grant # 266722 (SUMO project).

References

[1] Bakker, R., Schouten, J., Giles, C., Takens, F., Bleek, C.: Learning chaotic attractors by neural networks. Neural Computation **12**(10), 2355–2383 (2000)

[2] van den Berge, L.A., Selten, F.M., Wiegerinck, W., Duane, G.S.: A multi-model ensemble method that combines imperfect models through learning. Earth System Dynamics **2**(1), 161–177 (2011)

[3] Bishop, C.: Pattern recognition and machine learning. Springer (2006)

[4] Duane, G.: Synchronicity from synchronized chaos (2011). Arxiv.org/abs/1101.2213. Submitted

[5] Duane, G., Tribbia, J., Kirtman, B.: Consensus on Long-Range Prediction by Adaptive Synchronization of Models. In: D. N. Arabelos & C. C. Tscherning (ed.) EGU General Assembly Conference Abstracts, *EGU General Assembly Conference Abstracts*, vol. 11, p. 13324 (2009)

[6] Duane, G., Yu, D., Kocarev, L.: Identical synchronization, with translation invariance, implies parameter estimation. Physics Letters A **371**(5-6), 416 – 420 (2007)

[7] Eckmann, J.P., Ruelle, D.: Ergodic theory of chaos and strange attractors. Rev. Mod. Phys. **57**, 617–656 (1985)

[8] Frøyland, J., Alfsen, K.H.: Lyapunov-exponent spectra for the lorenz model. Phys. Rev. A **29**, 2928–2931 (1984)

[9] Houtekamer, P.: Variation of the predictability in a low-order spectral model of the atmospheric circulation. Tellus A **43**(3), 177–190 (1991)

[10] Kirtman, B., Min, D., Schopf, P., E., S.: A new approach for coupled gcm sensitivity studies. Tech. Rep. 154, COLA (2003)

[11] Kocarev, L., Shang, A., Chua, L.: Transition in dynamical regimes by driving: A unified method of control and synchronization of chaos. International Journal of Bifurcation and Chaos **3**(3), 479–483 (1993)

[12] Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. Advances in neural information processing systems pp. 231–238 (1995)

[13] Lapedes, A., Farber, R.: Nonlinear signal processing using neural networks: Prediction and system modelling. In: IEEE international conference on neural networks (1987)

[14] Lorenz, E.: Deterministic nonperiodic flow. Atmos J Sci **20**, 130–141 (1963)

[15] Lorenz, E.: Irregularity: a fundamental property of the atmosphere*. Tellus A **36**(2), 98–110 (1984)

[16] Mirchev, M., Duane, G.S., Tang, W.K., Kocarev, L.: Improved modeling by coupling imperfect models. Communications in Nonlinear Science and Numerical Simulation **17**(7), 2741 – 2751 (2012)

[17] Monteleoni, C., Schmidt, G.A., Saroha, S., Asplund, E.: Tracking climate models. Statistical Analysis and Data Mining **4**(4), 372–392 (2011)

[18] Olfati-Saber, R., Fax, J., Murray, R.: Consensus and cooperation in networked multi-agent systems. Proceedings of the IEEE **95**(1), 215–233 (2007)

[19] Pecora, L., Carroll, T.: Synchronization in chaotic systems. Physical review letters **64**(8), 821–824 (1990)

[20] Perrone, M., Cooper, L.: When networks disagree: Ensemble methods for hybrid neural networks. In: R. Mammone (ed.) Artificial neural networks for speech and vision, p. 126. Chapman & Hall (1994)

[21] Principe, J., Kuo, J.: Dynamic modelling of chaotic time series with neural networks. In: Advances in Neural Information Processing Systems, 7. Citeseer (1995)

[22] Principe, J., Rathie, A., Kuo, J.: Prediction of chaotic time series with neural networks. Nonlinear dynamic of the brain (Editor B. Jansen) pp. 250–258 (1993)

[23] Selten, F.: Toward an optimal description of atmospheric flow. Journal of the atmospheric sciences **50**(6), 861–877 (1993)

[24] Solomon, S., Qin, D., Manning, M., Chen, Z., Marquis, M., Averyt, K., Tignor, M., Miller, H.: Ipcc, 2007: Climate change 2007: The physical science basis. contribution of working group i to the fourth assessment report of the intergovernmental panel on climate change (2007)

[25] Sun, J., Bollt, E., Nishikawa, T.: Master stability functions for coupled nearly identical dynamical systems. Europhysics Letters **85**, 60,011 (2009)

[26] Tebaldi, C., Knutti, R.: The use of the multi-model ensemble in probabilistic climate projections. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **365**(1857), 2053 (2007)

[27] Wiegerinck, W., Selten, F.: Supermodeling: Combining imperfect models through learning. In: NIPS Workshop on Machine Learning for Sustainability (MLSUST) (2011). URL `http://people.csail.mit.edu/kolter/mlsust11/lib/exe/fetch.php?media=wiegerinck-mlsust.pdf`

[28] Yang, S., Baker, D., Li, H., Cordes, K., Huff, M., Nagpal, G., Okereke, E., Villafañe, J., Kalnay, E., Duane, G.: Data assimilation as synchronization of truth and model: Experiments with the three-variable lorenz system. Journal of the atmospheric sciences **63**(9), 2340–2354 (2006)

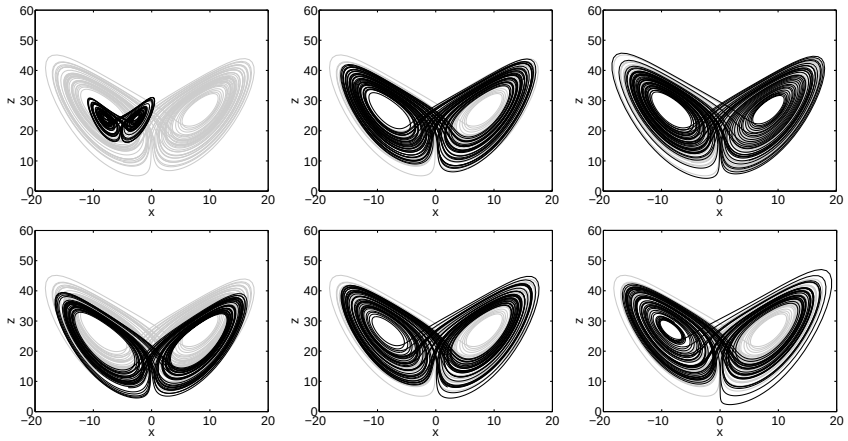

Figure 4: Lorenz 63. Left top: Uncoupled supermodel(a posteriori ensemble averaging). Left bottom: unweighted averaging supermodel. Middle top: connected supermodel, nudging. Middle bottom. Connected supermodel, cost function. Right top: weighted supermodel, cost function. Right bottom: weighted supermodel, quadratic programming. Supermodel (black) and ground truth (gray). The supermodel trajectories are based on a single run of $T = 100$ time units starting on the ground truth attractor, of which only the last $T = 50$ time units are plotted.

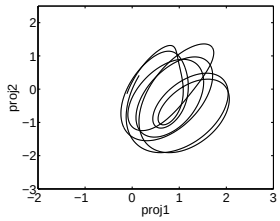

Figure 5: Lorenz 84. Trainingset of $T = 10$ time units generated by the assumed ground truth. The graph is a projection on the proj1 $= \sqrt{\frac{1}{2}}(x - y)$ and proj2 $= \sqrt{\frac{1}{6}}(-x - y + 2z)$ plane.

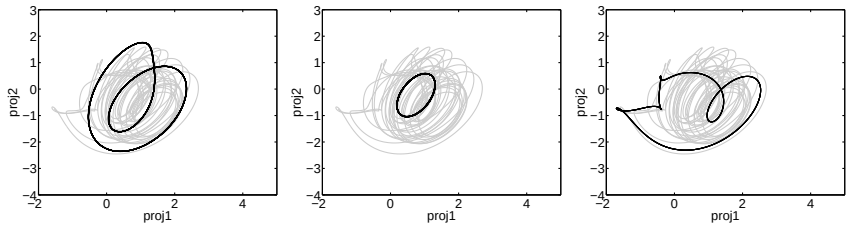

Figure 6: Lorenz 84 trajectories for the three connected imperfect models with connections determined by the learning process (black) and the standard Lorenz 84 system (grey).

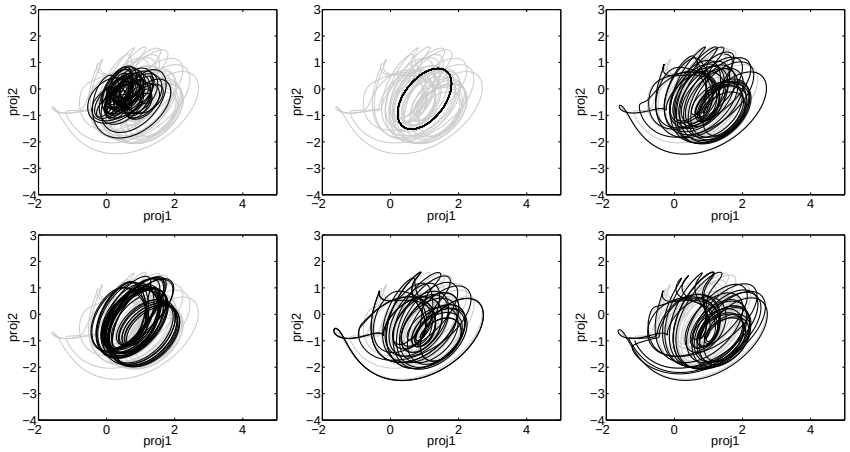

Figure 7: Lorenz 84. Left top: Uncoupled supermodel(a posteriori ensemble averaging). Left bottom: unweighted averaging supermodel. Middle top: connected supermodel, nudging. Middle bottom. Connected supermodel, cost function. Right top: weighted supermodel, cost function. Right bottom: weighted supermodel, quadratic programming. Supermodel (black) and ground truth (gray). The supermodel trajectories are based on a single run of $T = 100$ time units starting on the ground truth attractor, of which only the last $T = 50$ time units are plotted.

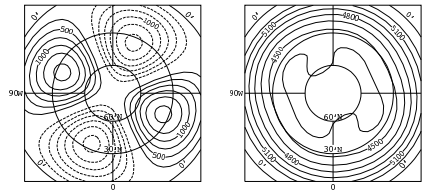

Figure 8: The topography (left) and the barotropic part of the climate (right) of the T5 model in a stereographic projection (geopotential height at 500 hPa). The contour interval in (a) is 250 m, in (b) 300 m.

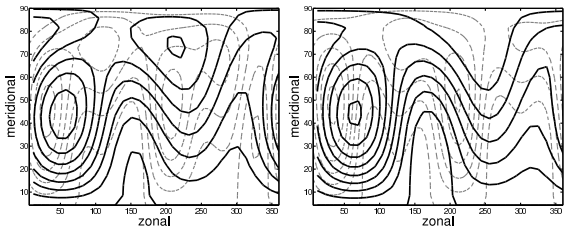

Figure 9: T5 model: the stream function at two subsequent times t and $t+5$ (days) at the 750 (gray, dashed) and 250 hPa (black, heavy) level. Wavy disturbances are visible, they grow and decay and travel eastward.

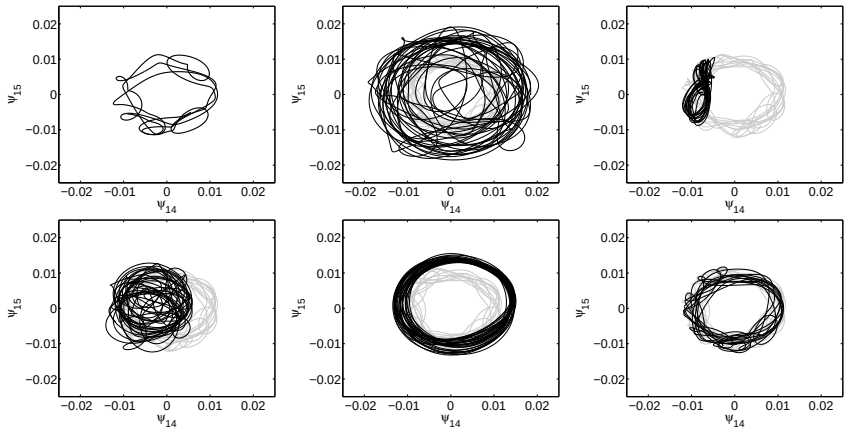

Figure 10: Results of T5 model integrations of the imperfect models projected on (ψ_{14}, ψ_{15}) plane. Top plots, left: training set. Top plots, middle and right: the imperfect models. Bottom plot, left: "unconnected ensemble". Bottom plot, middle"unweighted averaging". Bottom plot, right: optimization via quadratic programming. In all plots except the top left plot - Gray: assumed ground truth. Black: imperfect models and supermodels respectively.

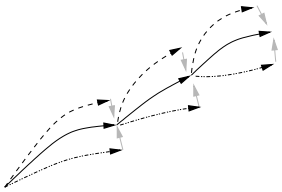

Figure 11: Representation of dynamics of a supermodel with limited information exchange. The dashed and dashed-dotted lines are trajectories of individual models, while the solid line is their weighted average, i.e. supermodel trajectory. The red arrows represent resets of individual models to supermodel states.