U. S. Department of Commerce
National Oceanic and Atmospheric Administration
National Weather Service
National Centers for Environmental Prediction

Office Note 476

FAST NN EMULATION OF THE SUPER-PARAMETERIZATION IN THE MULTI-SCALE MODELING FRAMEWORK

Vladimir M. Krasnopolsky[1,2], Michael S. Fox-Rabinovitz[2], Philip J. Rasch[3], and Minghuai Wang[3]

[1]Environmental Modeling Center, NCEP/NWS, NOAA

[2]Earth System Sciences Interdisciplinary Center, University of Maryland

[3]Pacific Northwest National Laboratory

January 2014

MMAB Contribution No. 316
e-mail: Vladimir.Krasnopolsky@NOAA.gov

Abstract

Extension of the NN emulation framework for superparameterization (SP) has been developed. The current PNNL-MMF has been used to provide the training set for a NN emulator. The large scale fields from the MMF that provide the SP inputs are used as the NN input vector and the aggregated results of the SP are used as the output vector. The limited training dataset was drawn selectively from four days of MMF global simulations, spanning diurnal and latitudinal variability. Experiments with different NN architectures have been performed and assessed. These experiments include comparisons of a single emulating NN with many outputs vs. a battery of simpler emulating NNs with different outputs. Also experiments with different output normalizations and different emulating NN complexities have been conducted. Training an ensemble of emulating NNs or developing NN-SP has been performed. Performance (accuracy and speedup) of the NN-SP has been estimated on an independent simulated dataset. Multiple statistics have been calculated for NN-SP and the differences between SP and NN-SP. The results obtained in this work can be summarized as follows: (1) The SP can be emulated by NN with a satisfactory accuracy (based on validation on an independent data set). (2) The NN-SP provides an impressive, *two orders of magnitude*, speedup as compared with the original SP. It provides a practical opportunity to use NN-MMF for decadal and longer climate simulations.

1. Introduction

Tremendous developments in numerical modeling and computing capabilities during the last decades have contributed dramatically to scientific and practical significance of climate, climate change, and weather prediction. One of the main problems of development and implementation of state-of-the-art models is the complexity of physical processes involved, especially at cloud scales. Parameterizations of model physics are approximate schemes, adjusted to model resolution and computer resources, based on simplified 1-D physical process equations and empirical and observational data. Still, calculations of model physics in a typical moderate resolution GCM (General Circulation Model) like the NCAR (National Center for Atmospheric Research) CAM-3 (Community Atmospheric Model) T42 (~3 degree) with 26 vertical levels, take about 70% of the total model computations.

A new approach to the treatment of model physics introduces cloud-scale processes into climate models by using Cloud-Resolving Models (CRM) nested into a conventional climate model (Grabowski 2001, 2002, 2003a,b, 2004, Randall et al. 2003, Khairutdinov and Randall 2001, 2003). This approach, originally labeled as a "superparameterization" (SP), but later as the Multi-scale Modeling Framework (MMF), is attracting a growing interest in the climate modeling community because it couples cloud-scale and large-scale dynamics within a single modeling framework. However, this approach imposes considerably more severe computational requirements because its inclusion increases the model run time by a factor of 200 to 250, which severely limits its applicability. To address this problem we propose to develop accurate and very fast neural network emulations of SP.

During the last decade, statistical or machine learning and specifically neural network (NN) techniques have found a variety of applications in different fields, including atmospheric and oceanic modeling (Krasnopolsky, 2007, 2013). The hybrid modeling approach has been introduced that is a synergetic combination of statistical (NN) and deterministic modeling. This approach uses NN for accurate and fast emulation or approximation of components of GCM model physics. Due to the capability of modern NN techniques to provide very high accuracy for approximation of complex multidimensional mappings like model physics, these NN emulations are practically identical to the original physical parameterizations themselves. In addition, they are orders of magnitude faster (Krasnopolsky et al. 2002, 2008, 2010).

Thus, NN emulations can preserve the integrity, accuracy, and, therefore, the level of sophistication of the state-of-the-art parameterizations of physical atmospheric processes. This result is achieved by using an atmospheric model run with the original parameterization to simulate training data for developing NNs. The model-simulated fields allow us to achieve the unprecedented accuracy for approximation because simulated data are free or almost free of the problems typical for empirical data (problems like high level of observational noise, sparse spatial and temporal coverage, poor representation of extreme events, etc.). In the context of our approach, the accuracy and speed-up of NN emulations is always measured against the original parameterization.

Here a feasibility or proof of concept study is presented for a novel application of the NN approach aimed at improving the overall performance of the MMF. Application of the NN approach has the capability of increasing the speed of calculations of the MMF to such a degree that would make it comparable with the speed of conventional climate models. This would, in turn, make the MMF more usable for climate studies and research on MMF improvements.

The central question of this paper is whether and how the SP (i.e., CRM embedded in the MMF) can be accurately approximated by a NN emulator. Following our previous work, we use an existing SP run inside the MMF for producing SP inputs and outputs needed for creating a NN training set. These simulated data come from a version of the MMF that treats multi-scale aerosol-cloud interactions, developed at Pacific Northwest National Laboratory (PNNL-MMF) (Wang et al., 2011). This version consists of a 2D CRM running inside each grid column of NCAR Community Atmosphere Model version 5 (NCAR CAM5) at 1.9 × 2.5 degree horizontal resolution, forced by climatological sea surface temperature (SST) values. The simulated data set of SP inputs and outputs is used to train the NN emulation of SP and to evaluate the NN emulation.

Before proceeding to descriptions of research background and methodology, three important issues should be briefly outlined. First, our NN technique is designed to produce NN emulations that are an accurate approximation of the fundamental physics implemented in the SP. By this we mean that the RMS errors or deviations of the quantities produced by the NN emulations from the same quantities produced by the existing SP are small and not growing in time. The RMS differences, for example, should be less than the magnitudes of typical observational errors. At the same time, the obtained biases have to be negligible. This means that the developed NN emulations have practically zero systematic errors and small RMS errors include only random errors. The research related to the use of the Monte Carlo Independent Pixel Approximation (Pincus et al. 2003) has shown that a climate model performance is insensitive to such errors in parameterizations. In some cases discussed below, closeness of statistical metrics of the data like mean value, variance, probability density function may be more important/indicative than small RMSE.

Secondly, in this work a NN emulation of the embedded SP is developed. This means that our NN emulation is, at best, only as good as the SP. It is our goal to make the NN emulation very close to the actual SP in terms of physical output, but computationally much faster. The NN emulation then replaces the SP in the MMF to enable longer model runs at much smaller computational cost and ensemble climate simulations, which may be computationally prohibited with MMF otherwise. Evidently, it does not, however, replace the SP in runs used to simulate training data.

The third point follows directly form the second. In this feasibility study only the existing SP is approximated using NN emulations. If this approach is shown to be feasible, improvements of the existing SP can be considered, including increasing the horizontal and vertical resolution and even introducing SP based on 3D CRM, which if emulated with NN becomes computationally affordable.

In Section 2 the background of MMF and that of the NN approach are introduced. In Section 3 SP inputs and outputs are identified. The simulated data used in this study are described in Section 4. In Section 5 details of the NN development are presented. In Section 6 the results of validation of NN emulation of SP (NN-SP) on independent data set, and an estimate of the speedup due to using NN emulation are presented. Conclusions are presented in Section 7.

2. Background

2.1 The PNNL Multi-scale Modeling Framework (MMF)

The PNNL-MMF has been described in detail in Wang et al. (2011), and is an extension of the Colorado State University (CSU) MMF model (Khairoutdinov and Randall, 2001). The MMF consists of a global climate model with a 2D cloud resolving model nested into each GCM grid square. The conventional cloud parameterizations are all removed from the model and the radiation code interacts directly with the CRM output. The MMF is not just a convection parameterization. MMF is also a turbulent boundary layer parameterization. The current host model uses the NCAR CAM5 model at 1.9 × 2.5 degree horizontal resolution and 30 vertical levels. The time step of the outer grid is 10 minutes. The host model has 13,824 grid cells, each of which contains a 2D CRM, which is nominally oriented north-south. The CRM uses non-hydrostatic, anelastic equations and a bulk condensed water formulation for cloud water, cloud ice, rain, snow and graupel. In the PNNL-MMF, the embedded CRM includes 64 columns with 4 km grid-spacing and 28 vertical layers coinciding with the lowest 28 CAM5 levels. The time step for the embedded CRM is 20 s. The CRM uses periodic boundary conditions. No information passes from one CRM to the next directly, but only through the mean flow of the large gird.

The CRM is called at each CAM time step. Each CRM retains its computed fields from the last time step as the initial fields for the current time step. The CRM equations are integrated continuously for the CAM time step and are forced by the large-scale tendencies computed as follows:

$$\left[\frac{\partial \Phi}{\partial t}\right]_{LS} = \frac{(\Phi_{LS} - \Phi_n)}{\Delta t_{LS}} \ , \tag{1}$$

where Φ is any CRM prognostic variable except precipitating water, Φ_{LS} is the corresponding variable computed by CAM as a result of all the large-scale processes after the MMF call at the previous CAM time step, Φ_n is the horizontally averaged CRM variable at the end of the CRM call at the previous CAM time step Δt_{LS}. This large-scale forcing represents *the relaxation of the CRM horizontal averages to the provisional CAM fields*. *The intent of this forcing term is to prevent a systematic drift of the CRM fields away from the corresponding large scale fields.* The CRM, in turn, returns the large-scale tendencies due to the cloud-scale processes computed as follows:

$$\left[\frac{\partial \Phi}{\partial t}\right]_{SP} = \frac{(\Phi_{n+1} - \Phi_{LS})}{\Delta t_{LS}} \tag{2}$$

where Φ_{n+1} is the horizontal mean of the CRM fields at the end of the CRM call. In the absence of cloud processes or convection resolved by the CRM domain, Φ_{n+1} will be identical to Φ_{LS} at the end of the MMF call, producing zero tendencies due to subgrid processes. As with the CSU simulations in 2D, no feedback occurs from the CRM to the large-scale wind because of concerns with the 2D momentum transfer.

Radiative transfer is computed interactively within the CRM domain and independently for each grid column, assuming 0-or-1 cloud fraction for each grid cell. The radiative transfer is computed every 10 minutes using the time averaged CRM fields. Previous research has demonstrated that cloud-scale interactions with the radiation are important for determining cloud evolution, the large-scale distribution of clouds, and cloud optical processes.

The PNNL-MMF is driven by the climatological sea surface temperature (SST). At the beginning of the simulation, the CRM columns were initialized with the CAM grid values and evolve continuously thereafter. Computational run time for the MMF is approximately 150 times slower than the parent CAM5

Changes to the current CRM configuration within the MMF are limited largely by computational overhead. In its 2D configuration at 4 km horizontal resolution, the model does a good job of capturing deep convection; hence, its improvements are demonstrated in simulations of the tropical Pacific climatology (Ovtchinnikov et al., 2005). However, the 4 km horizontal resolution is insufficient to resolve boundary layer clouds adequately. The ability of the CRM to simulate upper tropospheric cirrus is mixed, with some success in the case of more organized systems, but less so in the case of detached cirrus.

2.2 NN emulations of model physics

An ordinary model physics parameterization introduces into a model, in a parameterized form, an influence of subgrid scale processes, which the model does not resolve explicitly. It is a collection of equations and/or statistical relationships that, at each integration time step, take from the model a set of model variables, which constitute an input vector, X, and return to the model a set of variables, which constitute an output vector, Y. Usually the parameterization has no internal memory, that is, it does not use any information from the previous time steps to calculate the output vector Y. The vector Y provides the physical feedback or forcing used in the model to update model variables at the current integration step. Such a deterministic

parameterization for a particular input vector *X* generates a unique output vector *Y*. Radiation or convection parameterization can be considered as examples of such parameterizations.

The NN technique developed in our previous works (Krasnopolsky et al. 2002, 2008, and 2010) for emulating such deterministic parameterizations of model physics is based on two basic facts:

1. any deterministic parameterization of model physics can be considered as a single valued (unique output vector is produced for a particular input vector) continuous or almost continuous (like a step function) mapping (output vector, *Y*, vs. input vector, *X*, relationship),

$$Y = P(X) \tag{3}$$

2. NN (the multilayer perceptron in our case), which is a generic deterministic mapping, is a generic tool for approximation of such mappings. The deterministic mapping (3) can be emulated by a single NN (Chen and Chen 1995, Ripley 1996, Attali and Pagès 1997, Cherkassky and Mulier 1998).

The multilayer perceptron NN is an analytical approximation that uses a family of functions like:

$$y_q = a_{q0} + \sum_{j=1}^{k} a_{qj} \cdot \phi(b_{j0} + \sum_{i=1}^{n} b_{ji} \cdot x_i); \quad q = 1, 2, \ldots, m \tag{4}$$

where x_i and y_q are components of the input and output vectors respectively, a and b are fitting parameters, and ϕ is a so called activation function (usually it is a hyperbolic tangent), n and m are the numbers of inputs and outputs respectively, and k is the number of neurons in the hidden layer.

During last several decades the NN emulation technique has found the variety of applications in different fields (Krasnopolsky 2007, 2013) and, more specifically, for accurate and fast emulating atmospheric radiative processes and radiation parameterizations in GCMs (Krasnopolsky et al. 2008, 2010).

Based on our experience with NN emulations of deterministic parameterizations, a high accuracy for NN emulations can be obtained, with practically zero biases and small random errors. This is a necessary condition for successful integration of the NN emulations into a model, with preservation of the integrity/quality of the model physics and avoiding accumulating errors during long-term climate simulations with developed NN emulations. The choice of an optimal NN emulation is based on its accuracy not its speed-up that will be very significant, anyway.

2.3 Extension of the NN emulation developmental framework/methodology for SP

2.3.1 Approach

In this study, our goals are to:

1. further extend the NN emulation methodology to emulate SP,
2. develop a NN emulation of SP to significantly improve the model computational performance, test the approximation accuracy of the developed emulation on an independent test data set

Our approach takes advantage of the facts that:

1. in each column of the GCM at each time step the SP or the embedded CRM receives a vector of input parameters X, which describes a state of the atmosphere in this column in

terms/variables of the GCM (Φ_{LS} in eqs. (1)) and serves as a forcing for the embedded CRM;

2. after integration of the CRM in the column of the GCM, the SP returns back to the GCM a vector of output parameters *Y*, which describes the physical forcing for this column in terms/variables of the GCM (see eq. (2)).

Thus, the entire SP, from a mathematical point of view, can be considered as a mapping similar to (3). Taking into account the physical and mathematical properties of the CRM, this mapping is continuous or almost continuous (may contain finite discontinuities like step functions). However, there is a significant difference between a deterministic (e.g., radiation) parameterization, which can be adequately represented by the mapping (3), and the SP. This difference, which requires us to modify a mapping representing SP and our NN emulation technique, is discussed in the next section.

2.3.2 Memory/history dependence aspects of SP

In the case of SP, at each time step, the output of the mapping, representing SP, *Y*, depends not only on the SP input vector *X*, but also on an internal hidden (from GCM) variable set, which is the state of the CRM, ξ at the previous timestep. One can imagine the same GCM forcing *X* producing different results *Y* based on a different internal state of the CRM, ξ, which is not known to GCM and therefore to NN, which we are going to use for emulating SP. Actually, the SP, for the same *X*, may return to GCM various *Ys*, depending on different scenarios of subgrid processes, which are determined by the CRM conditions ξ_S. The CRM internal state is the result of CRM integration at the previous time step, which is remembered in SP. Thus, when considered as a mapping, SP demonstrates an uncertainty of the output vector *Y* at a particular value of the input vector *X*, due to the variability of hidden internal CRM initial conditions. This

uncertainty has a physical meaning; variability or uncertainty of Y for the same X represents in GCM the subgrid variability, i.e. effects of subgrid processes. Thus, this memory/history dependency needs to be taken into account and the mapping (3) has to be modified for SP.

We can introduce formally/mathematically the history or memory dependence modifying for SP the mapping (3) as,

$$Y = SP(X, \xi), \tag{5}$$

where ξ is the vector of the initial state of the CRM. Let us stress again that ξ is a hidden variable not known to the GCM. It is an internal SP variable, which is not available outside of SP, in the GCM, and therefore it is not available for the NN emulating SP. Thus, the mapping (5), unlike mapping (3), is not a deterministic (single valued) mapping; it is a multivalued mapping, and the NN emulation approach developed for mapping (3) should be modified in this case.

Because ξ is not available in the GCM, it can be considered as a stochastic variable determined by processes that are not recognizable at GCM scales. Thus, mapping (5) can be considered as a stochastic mapping that is a family of mappings (or a multivalued mapping). The probability distribution of the members of the family is determined by X and ξ. A NN technique for emulating stochastic mappings has been developed in Krasnopolsky et al. (2011) and Krasnopolsky and Lin (2012) and was applied to emulate convection parameterization derived from data simulated by CRM (Krasnopolsky et al. 2011) and to evaluate a NN multi-model ensemble (Krasnopolsky and Lin 2012). The generic and flexible NN technique provides the necessary tools for emulating such mappings. An ensemble of NNs was introduced to emulate a stochastic mapping. There are many different methods that can be used to generate an ensemble

of NNs (e.g., see Krasnopolsky 2013). The most common approach is to train a set of NNs using different initializations for their weights. In this case, different members of the NN ensemble correspond to different local minima of the error function, which is minimized during the NN training.

Thus, the stochastic mapping (5), which actually is a family of mappings, is emulated by an ensemble (a family) of NNs:

$$Y_i = Net_i(X) = Net(X,W_i), \quad i = 1, \ldots, M \tag{6}$$

where each Net_i is represented by eq. (3), W_i is a set of a and b in eq. (3) corresponding to the local minimum number i of the error function, and M (see Section 5.2) is the number of the ensemble members. The ensemble (6) constitutes NN emulation for SP (5). An alternative approach to build NN emulation for SP exists and is briefly discussed in Section 7; however, in this study we use the NN ensemble (6) to emulate SP in MMF.

3. SP Inputs and Outputs

The definition of what are constituents of vectors *X* and *Y* is an essential part of the conceptual design of a NN emulation. While we have a current working understanding of these vector's components, this understanding will may have to be refined and re-determined experimentally in the process of the development.

The SP inputs are listed in Table 1 and SP outputs – in Table 2. Vertical dimensionalities of variables are shown for these SP inputs and outputs in Tables 1 and 2 and in the following tables because it is this dimensionality that determines the dimensionalities of the mapping (5) and eventually the size of the emulating NN (number of NN inputs and outputs).

Table 1 SP Inputs and Their Vertical Dimensionality

Input #	Physical Name	Variable	Vertical Dimensionality in GCM
1	Temperature	T3D	30
2	Water vapor	Q3D	30
3	Liquid cloud water	QC3D	30
4	Ice cloud water	QI3D	30
5	Horizontal U component of the velocity vector	U3D	30
6	Horizontal V component of the velocity vector	V3D	30
7	Vertical pressure velocity	OMEGA3D	30
8	Pressure depth	PDEL3D	30
9	Pressure at the middle of a model layer	PMID3D	30
10	Grid height at the middle of a model layer	ZMID3D	30
11	Grid height at model interface	ZINT3D	31
12	Radiative heating rate	RAD3D	30
13	Aerosol number concentrations (for 3 models)	num_a1_NUM	30
14		num_a2_NUM	30
15		num_a3_NUM	30
16	Aerosol volume concentrations	num_a1_VOL	30
17		num_a2_VOL	30

18		num_a3_VOL	30
19	Aerosol Hygroscopicity	num_a1_HYG	30
20		num_a2_HYG	30
21		num_a3_HYG	30
22	Surface pressure	PS2D	1
23	Large-scale surface stress	TAU2D	1
24	Large-scale surface wind	WNDLS2D	1
25	Large-scale surface buoyance flux	BFLXLS2D	1
26	Temperature from last step	T3DOLD	30
27	Water vapor from last step	Q3DOLD	30
28	Liquid cloud water from last step	QC3DOLD	30
29	Ice cloud water from last step	QI3DOLD	30
30	Rain water from last step	QR3DOLD	30
31	Snow water from last step	QS3DOLD	30
32	Graupel water from last step	QG3DOLD	30
33	Horizontal U component of the velocity vector from last step	U3DOLD	30
34	Large-scale temperature tendency	DT3DIN	30
35	Large-scale total water tendency	DQ3DIN	30
36	Large-scale zonal wind speed tendency	DU3DIN	30
37	Large-scale meridional wind speed tendency	DV3DIN	30
	Total Vertical Dimensionality		995

Table 2 SP Outputs

Output #	Physical Name	Variable	Vertical Dimensionality in GCM
1	Temperature tendency	SPDT	30
2	Water vapor tendency	SPDQ	30
3	Liquid water tendency	SPDQC	30
4	Ice water tendency	SPDQI	30
5	Cloud fraction	CLOUD	30
6	Surface precipitation rate	PRECT2D	1
7	Surface ice precipitation	PRECST2D	1
	Total Vertical Dimensionality		152

4. Data sets

To generate data for the development of NN emulation of SP, MMF was run from June 1st to July 10th. Since the model is driven by climatological SSTs, the model results do not correspond to any particular year. 3 hourly model output on June 20, 25, 30 and on July 5 and 10 were used in this study (45 time samples in total). Each time sample contains 144 × 96 = 13,824 horizontal grid points. At each of these grid points, profiles (30 vertical levels) of all SP inputs and outputs listed in Tables 1 and 2 are available. Almost all these data have been used to create the training, test, and validation data sets. The input variable number 11, "Grid height at model interface", which is a redundant variable (there is an almost identical variable "grid height at the middle of a model layer" there), was excluded. "Rain water", "Snow water", and "Graupel water" from the previous integration step have been also removed after testing as NN inputs and determining that they do not affect the accuracy of NN emulations. All other variables have been tested level by level and the levels with constant values have been removed from the corresponding profiles because they are not contributing to the SP input/output relationship learned by NN in the process of NN development. Three additional variables have been added to each input profile: latitude, longitude, and time of the day for each profile, which are important for learning the diurnal cycle.

Each created data set consists of records composed of input and output vectors. Each of these vectors is composed of profiles of input variables and output variables correspondingly. Constituent profiles and formats of input and output vectors are presented in Tables 3 and 4. These tables show: the initial dimensionality of each constituent profile, its dimensionality after removing levels with constant values (usually zero or a very small number), the indexes of the

remaining non-constant levels in the profile (a reduced profile), and the indexes of the reduced profile in the input/output vector.

Table 3 Format of the Input Vector in Data Sets

Input #	Physical Name	Variable	Initial Vert. Dim.	Reduced Vert. Dim.	Start:End indexes in profile	Start:End indexes in data set
1	Temperature	T3D	30	30	0:29	0:29
2	Water vapor	Q3D	30	30	0:29	30:59
3	Liquid cloud water	QC3D	30	17	13:29	60:76
4	Ice cloud water	QI3D	30	25	5:29	77:101
5	Horizontal U component of the velocity vector	U3D	30	30	0:29	102:131
6	Horizontal V component of the velocity vector	V3D	30	30	0:29	132:161
7	Vertical pressure velocity	OMEGA3D	30	30	0:29	162:191
8	Pressure depth	PDEL3D	30	18	12:29	192:209
9	Pressure at the middle of a model layer	PMID3D	30	18	12:29	210:227
10	Grid height at the middle of a model layer	ZMID3D	30	30	0:29	228:257
11	Radiative heating rate	RAD3D	30	28	2:29	258:285
12	Aerosol number concentrations (for 3 models)	num_a1_NUM	30	30	0:29	286:315
13		num_a2_NUM	30	30	0:29	316:345
14		num_a3_NUM	30	30	0:29	346:375
15	Aerosol volume concentrations	num_a1_VOL	30	30	0:29	376:405
16		num_a2_VOL	30	30	0:29	406:435
17		num_a3_VOL	30	29	1:29	436:464
18	Aerosol Hygroscopicity	num_a1_HYG	30	30	0:29	465:494
19		num_a2_HYG	30	30	0:29	495:524
20		num_a3_HYG	30	30	0:29	525:554
21	Temperature from last step	T3DOLD	30	28	2:29	555:582

22	Water vapor from last step	Q3DOLD	30	28	2:29	583:610
23	Liquid cloud water from last step	QC3DOLD	30	17	13:29	611:627
24	Ice cloud water from last step	QI3DOLD	30	25	5:29	628:652
25	Horizontal U component of the velocity vector from last step	U3DOLD	30	28	2:29	653:680
26	Large-scale temperature tendency	DT3DIN	30	28	2:29	681:708
27	Large-scale total water tendency	DQ3DIN	30	28	2:29	709:736
28	Large-scale zonal wind speed tendency	DU3DIN	30	28	2:29	737:764
29	Large-scale meridional wind speed tendency	DV3DIN	30	28	2:29	765:792
30	Surface pressure	PS2D	1	1	0:0	793:793
31	Large-scale surface stress	TAU2D	1	1	0:0	794:794
32	Large-scale surface wind	WNDLS2D	1	1	0:0	795:795
33	Large-scale surface buoyance flux	BFLXLS2D	1	1	0:0	796:796
34	time of day for the profile (from 0 to 7)	TIME2D	1	1	0:0	797:797
35	Lat for the profile	LAT2D	1	1	0:0	798:798
36	Lon for the profile	LON2D	1	1	0:0	799:799
	Total Vertical Dimensionality		**877**	**800**		

Table 4 Format of the Output Vector in Data Sets

Output #	Physical Name	Variable	Initial Vert. Dim.	Final Vert. Dim.	Start:End index in the profile	Start:End index in the set
1	Temperature tendency from SP	SPDT	30	26	4:29	0:25
2	Water vapor tendency from SP	SPDQ	30	26	4:29	26:51
3	Liquid water tendency from SP	SPDQC	30	18	12:29	52:69
4	Ice water tendency from SP	SPDQI	30	25	5:29	70:94
5	Cloud fraction	CLOUD	30	24	6:29	95:118
6	Surface precipitation rate	PRECT2D	1	1	0:0	119:119
7	Surface ice precipitation	PRECST2D	1	1	0:0	120:120
	Total Vertical Dimensionality		**152**	**121**		

Finally, 487,368 profiles created from 36 time samples, corresponding to June 20, 25, and 30 and July 5 were randomly split into two equal data sets: the training set and the test set (see Table 5). 9 time samples corresponding to July 10 were used to create the validation set (see Table 5).

Table 5 Data sets used for development of NN emulation of SP

	File Name	# of Profiles	# of Inputs	# of Outputs	Time Covered
Training Set	MMF-4-days_glob_800-121.1	243,684	800	121	4 days, 6/20, 25, 30, & 7/5
Test Set	MMF-4-days_glob_800-121.0	243,684	800	121	"
Validation Set	MMF-1-days_glob_800-121	121,842	800	121	One day, 7/10

5. Development of NN Emulations

5.1 NN Architecture

Selecting the NN architecture includes several steps. First, it is our initial hypothesis that one NN emulator can be constructed for all global conditions. Second, we select emulating NN inputs and outputs. Third, we decide if we use a single NN with multiple outputs or an array of simpler NNs with one output to emulate the mapping (5). Then, if we select using a single emulating NN with multiple outputs, several different approaches for output normalization are available. A particular normalization procedure has to be chosen. Finally, the NN complexity, that is the number of neurons in the hidden layer, k, has to be selected.

5.1.1 Inputs and Outputs

Not all SP inputs are necessarily to be included as inputs of emulating NNs. Only those that are important for emulating input/output relationship must be included. However, because we have only some qualitative knowledge based on physical intuition about importance of different inputs, we split all input variables into two classes: basic inputs and auxiliary inputs. In the first class 31 variables from the current GCM integration step (variable numbers from 1 to 20 and 26 to 36 in Table 3) are included; these variables are called "Basic" in Table 6. The second class consists of five variables (variable numbers from 21 to 25 in Table 3) providing information from the previous GCM integration step.

In this first attempt to develop a NN emulation of SP, we assumed that all basic variables are important to emulate SP input/output relationship; thus, we included them as inputs in all NN architectures considered below. All inputs are normalized to the interval [-1,1]. Then we experimented (included or removed) with auxiliary input variables only. The architectures

investigated here are presented in Table 6. All investigated NN architectures have 121 outputs, i.e. include all seven output variables (five 3D and two 2D) presented in Table 4.

For each of the presented architectures, for each output variable five error statistics (bias, RMSE, min error, max error, and correlation coefficient) have been calculated, 35 statistics totally per NN architecture. Then NNs were scored using these statistics. All NNs were scored for each of 35 statistics separately. Then for each NN all 35 scores were totaled and a total rank derived. The NNs were ranked accordingly (see the right column in Table 6). All architectures were compared with the base architecture #0, where the emulating NN includes all 36 input variables. The NN with the architecture #III (the winner) has all error statistics very close to the base configuration #0; however, it has 57 less inputs than the base architecture, which significantly reduces the training time for #III as compared with the #0. We would like to caution readers not to derive immediate conclusions about relative physical importance of various input variables based on the results of Table 6, as it is usually done with the linear regression results. The mapping (5) is a very complex nonlinear mapping and NN is a very complex nonlinear statistical approximation with significantly correlated inputs and outputs. In this situation a straightforward linear thinking is not a reliable tool.

Table 6 Different NN Architectures investigated in the study.

Architecture #	Input variables included	Rank
0	All 36 variables	
I	Basic[1] + #25(u3dold)	3
II	Basic + #23(qc3dold) + #24(qi3dold)	4
III	Basic + #23(qc3dold) + #24(qi3dold) + #25 (u3dold)	1
IV	Basic + #22(q3dold) + #23(qc3dold) + #24(qi3dold) + #25(u3dold)	5
V	Basic+ #21(t3dold) + #22(q3dold) + #23(qc3dold) + #24(qi3dold)	2

[1] Basic includes 31 variables numbers from 1 to 20 and 26 to 36 in Table 3.

For this feasibility study we selected the NN architecture #III for all following experiments. All NNs considered below, that is all ensemble members, have the same number of inputs (n in (4); n = 746) and outputs (m in (4); m = 121). The number of the inputs is 746 not 744 because two variables (TIME2D and LON2D) are converted into four cyclic variables: $\sin(\frac{2\pi \cdot TIME2D}{7})$, $\cos(\frac{2\pi \cdot TIME2D}{7})$, $\sin(\frac{2\pi \cdot LON2D}{360})$, and $\cos(\frac{2\pi \cdot LON2D}{360})$.

In MMF the GCM only requires a limited number (seven) of variables as output from the SP. The CRM inside SP, however, produces a much richer set of variables such, for example, as cloud fraction and optical depth, that are important diagnostics. We did not evaluate the fidelity of these output variables, but they could be included into the output vector Y.

5.1.2 NN complexity and calculation time

The number of NN weights, N_c, (Krasnopolsky 2013) can be used as a measure of the complexity of the emulating NN, which should correspond to the functional complexity of a mapping represented by (3) or (5),

$$N_c = k \cdot (n + m + 1) + m \tag{7}$$

However, when we compare performance of NNs with the different number of outputs, the number of NN weights per output, n_c, is a better measure to use as a more adequate estimate of the NN complexity per output,

$$n_c = \frac{N_c}{m} = k \cdot \frac{(n + m + 1)}{m} + 1 \tag{8}$$

Here n and m are the numbers of inputs and outputs respectively, and k is the number of neurons in the hidden layer. In addition to the functional complexity, n_c determines the numerical complexity of the NN emulation and the calculation time required to calculate one NN output.

5.1.3 Single NN vs. an array/battery of NNs and normalization of outputs

NN is a very flexible tool; it offers various options for an architecture of NN emulating a mapping represented by (3) or (5). These options should be considered and an optimal one should be selected taking into account physical properties of the mapping and numerical requirements for the NN emulation. Our mapping has n = 746 inputs and m = 121 outputs. We can choose to emulate it with a single NN that has n = 746 inputs and m = 121 outputs or with an array (a battery) of m = 121 NNs with n = 746 inputs and one output each. Many intermediate solutions are available; however, we will not consider them here, because comparisons of the two aforementioned cases already have led us to a clear conclusion.

To compare a single emulating NN with multiple outputs with the array of emulating NNs with one output each, we trained several NNs with multiple outputs and several NNs with a single output. We selected a couple of particular outputs (among 121 outputs) for training NNs with one output and for comparisons with the same outputs of a single NN with 121 outputs. As the first single output for NNs with the single output we selected the value of the temperature tendency, *spdt*, at the lowest (closest to the ground) level, where the magnitudes of *spdt* and of its errors are maximal. The reason that the tendencies are largest at the surface is because of the deposition of heat into the surface layer from the surface fluxes.

We trained several NNs with n = 746 inputs and one output each and with k = 1, 2, 3, and 4 in one hidden layer. Then, for each trained NN, we calculate error statistics for the single output and NN complexity, using eq. (8) with n = 746 and m = 1. Also we trained several emulating NN with n = 746 inputs, m = 121 outputs, and with k = 25, 50, 75, 100, 125, and 150 each. NN outputs have been normalized using the normalization (10) defined below. For each of these emulating NNs with multiple outputs, we calculated statistics for the same single output, for which NNs with a single output have been trained (*spdt* at the lowest, closest to the ground, level). Also for these NNs the complexity per output, n_c, have been calculated, using eq. (8) with n = 746 and m = 121. Fig. 1 presents the comparison of NN approximation RMSEs for *spdt* at the lowest, closest to the ground, level, for all aforementioned emulating NNs.

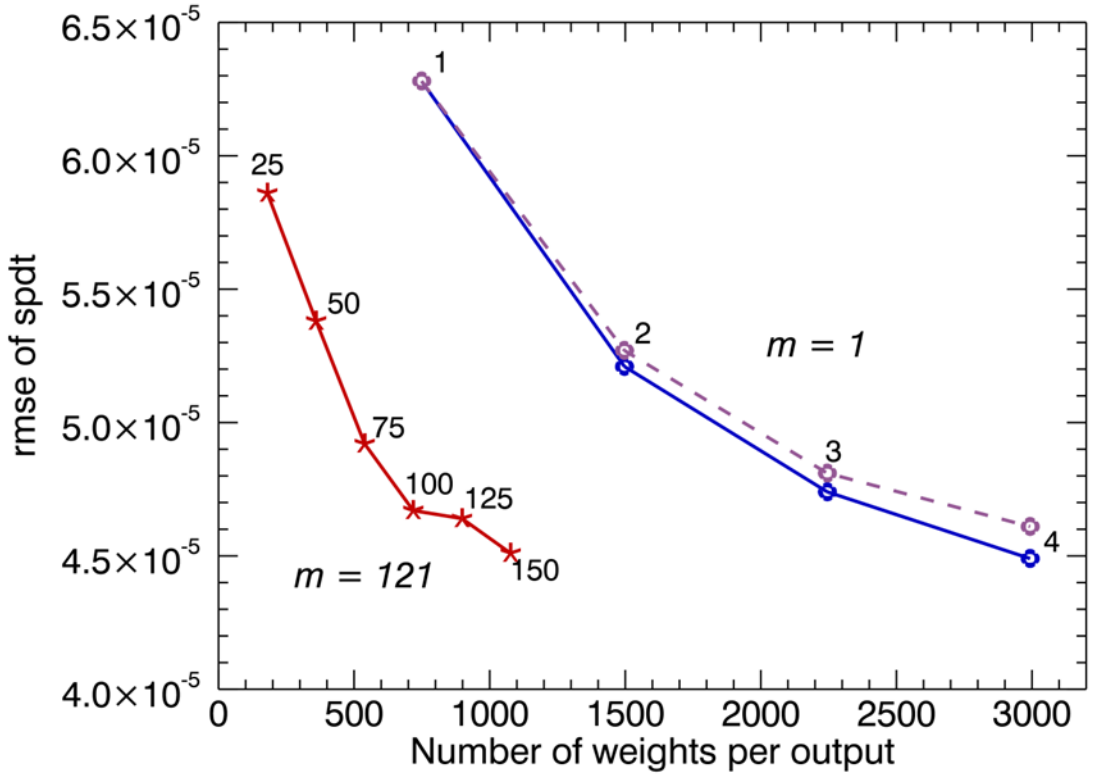

Fig.1 NN approximation RMSEs (calculated on independent test set) for a single output, *spdt* at the lowest vertical level vs. NN complexity per output, n_c. Emulating NNs with n = 746, m = 121 (all NNs use the output normalization (10)) and various number of hidden neurons, k (shown with numbers) are represented by the red solid line with asterisks. The NNs with n = 746, m = 1 and various number of hidden neurons, k (shown with numbers) are represented with the violet dashed line with circles. The blue solid line with circles shows results for RMSEs for NNs with n = 746, m = 1 and various number of hidden neurons calculated on the training set, for comparison.

At the first sight, the results presented at Fig. 1 may contain two seeming "paradoxes". First, the complexity per output of a NN with one output and two neurons in one hidden layer is greater than that of a NN with 121 outputs and 150 hidden neurons in one hidden layer. Formally, this result can be verified using eq. (8); however, to understand it better, we should analyze eq. (4). As can be easily concluded from this equation, for the NN with *any number of outputs*, outputs are built from the same k building blocks (basis functions or hidden neurons), ϕ. All k basis functions must be calculated, independently of the number of outputs, for NN with any number of outputs. This operation gives the major contribution to NN complexity and computation time.

Then, in the case of multiple outputs, calculating m linear combinations of basis function to obtain m outputs does not add significantly to the NN complexity and calculation time (as compared with m = 1), reducing at the same time the complexity (and calculation time) per output (as compared with a single output NN) due to m in the denominator of eq. (8).

The second "paradox" in the results depicted in Fig. 1 is that a single NN with multiple outputs provides a significantly higher accuracy of approximation than a NN with a single output with the same complexity per output, n_c. Here we should notice that the SP outputs are not completely independent; there are physical relationships between them defined by imbedded CRM physics. The SP outputs satisfy these relationships with high accuracy because the relationship is explicitly (or implicitly) included into the imbedded CRM equations. Thus, these relationships are implicitly imbedded in the output data presented to NN in the training set as correlations between outputs. There are at least two different types of such correlations. The first type is the correlations due to the aforementioned physical relationships between different output variables. Also, some of the output variables are vertical profiles. Thus, the second type of correlations between outputs is the correlations between adjacent components of the vertical profiles that constitute the output vector. These correlations are due to the fact that vertical profiles are discretized continuous functions. Both types of the aforementioned correlations can constitute a significant and irreplaceable (see Fig.2 and the discussion below) part of information about mappings (3 or 5) containing in the training set.

Obviously, when a NN emulation with a single output (m = 1) is trained, this part of information is completely lost, which explains why the accuracy of the NN emulation with multiple outputs is significantly better, for a selected single output, than the accuracy of the same output when it is emulated by a NN with this single output with the same complexity per output. *Thus, for*

multidimensional mappings, it is preferable to use a single emulating NN with multiple outputs than an array of NNs with a single output, both in terms of the approximation accuracy and the computational performance.

Obviously, even if we train a single NN with multiple outputs, the outputs of the NN emulations satisfy the aforementioned relationships only approximately but with high accuracy. If even higher accuracy of these relationships is desired (e.g., for conservation relationships), some constraints may be formulated in the *Y* (output vector) terms and can be taken outside of the NN emulation and imposed/included there as a post-processing step. In this case, they are applied to the NN output outside, i. e., in GCM. Note that we have similar situations with the long and short wave radiation parameterizations. Their outputs (heating rates and fluxes) are related. To force NN outputs to satisfy this relationship exactly, a balancing procedure has been developed (Krasnopolsky et al. 2010), which is applied to NN outputs as a post-processing step. Also, some constraints may be included in NN or applied in the process of the NN training.

A choice of a normalization of outputs is closely related to our choice of a single NN with multiple outputs vs. an array of NNs with a single output each. For NN with a single output the accuracy practically does not depend on the type of the output normalization (Krasnopolsky 2013). For a single NN with multiple outputs at least two different types of normalization can be introduced. The multiple outputs can be normalized independently; in this case the q^{th} output can be normalized as,

$$y_q' = \alpha \cdot \frac{y_q - \bar{y}_q}{\sigma_q} \tag{9}$$

where $\bar{y}_q$ and σ_q are the mean and SD of the q^{th} output, y_q, and $\alpha \leq 1$ is introduced to accelerate the training of the linear weights a in the output layer of the NN (4). This normalization

improves approximation accuracies for small outputs; however, if these outputs are noisy, it propagates the noise to other outputs. Normalization (9) also reduces or even destroys correlations that may exist between outputs.

An alternate normalization is,

$$y_q' = \alpha \cdot \frac{y_q - \bar{y}_q}{\sigma} \tag{10}$$

where σ is the SD for all outputs or for a group of L outputs,

$$\sigma = \sqrt{\frac{1}{(P \cdot L - 1)} \sum_{q=1}^{L} \sum_{i=1}^{P} \left(y_q^i - \bar{y} \right)^2} \tag{11}$$

where P is the number of records in the training set and $\bar{y}$ is the mean value of L outputs in the training set. The normalization (10) normalizes the entire group of L outputs (a profile) coherently; thus, it preserves all correlations between outputs in the group. Thus, each profile contributing in Y is normalized independently. The normalization (10) improves the accuracy of larger outputs because it enhances their contribution in the error function, which is minimized during the NN training.

Fig. 2 presents the comparison of NN approximation RMSEs for *spdqc* at the vertical level #11. At this level *spdqc* reaches a maximal value. Here we used the same six emulating NN with n = 746 inputs, m = 121 outputs, and with k = 25, 50, 75, 100, 125, and 150 each, with output normalized using (10) (the red solid line with asterisks). In addition we trained five single emulating NN with n = 746 inputs, m = 121 outputs, and with k = 50, 75, 100, 125, and 150 each, with output normalized using (9) (the pink solid line with squares). For comparison, the results for four NNs with the single output are presented here also (the violet dashed line with circles).

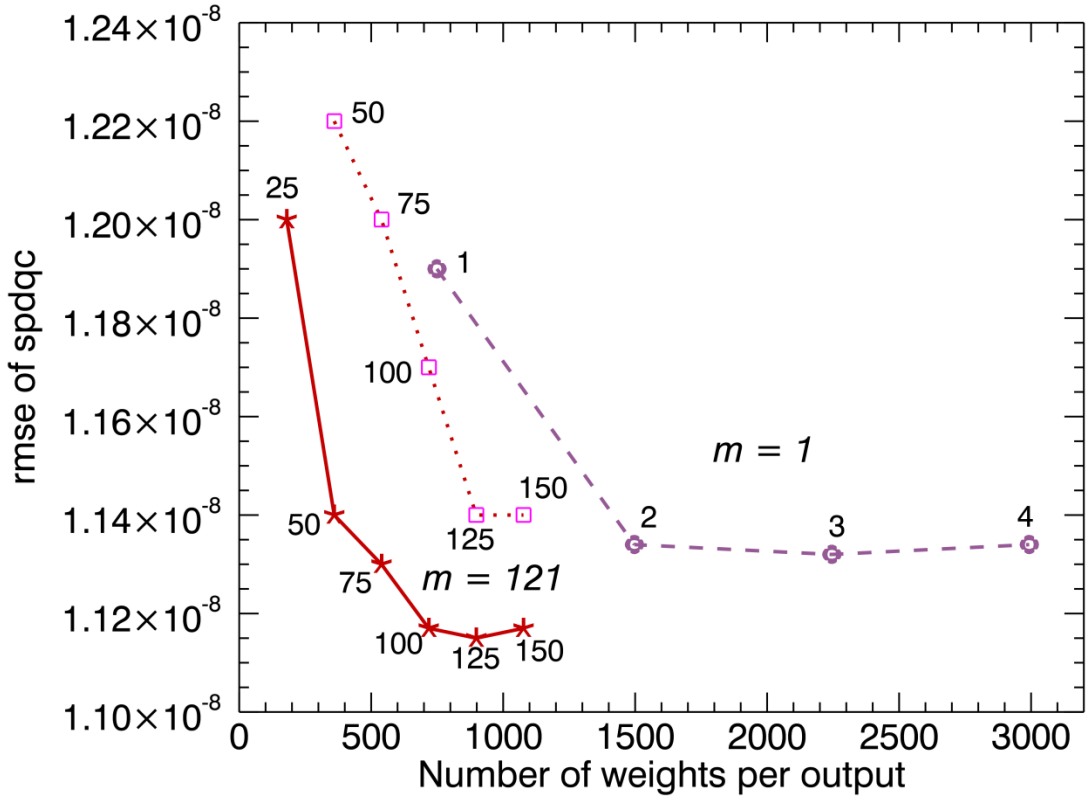

Fig.2 NN approximation RMSEs (calculated on independent test set) for a single output, *spdqc* at the vertical level #11 vs. NN complexity per output, n_c. Emulating NNs with n = 746, m = 121 (all NNs use the output normalization (10)) and various number of hidden neurons, k (shown with numbers) are represented by the red solid line with asterisks. The pink solid line with squares shows RMSEs for emulating NNs with n = 746, m = 121 (all NNs use the output normalization (9)) and various number of hidden neurons. The NNs with n = 746, m = 1 and various number of hidden neurons, k (shown with numbers) are shown with the violet dashed line with circles for comparison.

Fig. 2 shows, that NNs using normalization (10) can take significant advantage of substantial information about correlations between outputs available in the training set. This information is really important and significant. Fig. 2 shows that NNs using normalization (9), which partly destroys this information or NNs with a single output, may never reach such approximation accuracy as NNs using normalization (10). For the variables like *spdqc*, which demonstrate high level of stochasticity and uncertainty, a proper normalization of outputs is essential. It allows NN to use an additional information about the mapping to be emulated which is imbedded in an

output vector as correlations between the vector components and, thus, to reduce the output uncertainty.

Finally, based on the aforementioned results, the seven output variables have been separated in two groups, in terms of the normalization. Five vector variables (profiles) (see Table 4) have been normalized using (10), each profile separately. Two scalar surface variables have been normalized separately using (10), which in this case is equivalent to (9). All following results have been obtained with NNs, which use this normalization of outputs. This normalization reduces the complexity and improves the performance of the emulating NN.

5.1.4 Number of Hidden Neurons

In previous sections we selected a single NN with multiple outputs to emulate SP. We also selected inputs and outputs and the normalization of outputs. The only parameter of the NN architecture that remains to be chosen is the number of hidden neurons, *k*, which determines the emulating NN complexity. Ideally, the NN complexity should correspond to the mapping (5) functional complexity (Krasnopolsky 2013). However, because there are no mathematical tools developed to estimate nonlinear mapping complexities, *k*, is usually determined empirically.

We have trained six emulating NNs with n = 746 inputs, m = 121 outputs, and with k = 25, 50, 75, 100, 125, and 150 each. The corresponding values of n_c are shown in Table 7. The convergences of the approximation RMSEs are shown in Fig. 3 for four prognostic outputs (profiles) of SP (defined in Table 4): *spdt*, *spdq*, *spdqc*, and *spdqi*. Blue lines show RMSEs calculated using the training set and red – the independent test set. RMSEs are shown vs. the NN complexity per output, n_c. Fig. 3 shows RMSEs for six trained NN mentioned above (shown with asterisks).

Table 7 The NN complexity per output (the number of weights per output), n_c, corresponding to various numbers of hidden neurons *k*.

k	25	50	75	100	125	150
$\boldsymbol{n_c}$	180	360	539	718	898	1077

RMSE is calculated for the entire output profile as,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{P}\sum_{j=t+1}^{t+L}\left[Y_i^j - Y_{i\,NN}^j\right]^2}{P \times L}} \tag{12}$$

where Y_i^j and $Y_{i\,NN}^j$ are outputs from the original SP and its NN emulation, respectively, the index $i=1,\ldots, P$ determines the record number in the validation data set, P is the number of the records,

and $j = t+1,\ldots,t+L$ is the vertical index of the corresponding profile in the SP output vector (t is the index of the first profile component in the output vector Y) where L is the number of the profile vertical components.

As can be concluded from Fig. 3, for $k > 50$, the RMSEs calculated on the independent test set do not improve (for *spdqi*), improve insignificantly(for *spdqc*), or improve very little (less than 8% for *spdt* and *spdq*). Because k (and n_c) determines the NN estimation time and, more importantly, the NN training time, in our first attempt to emulate SP, we selected $k = 50$ as a reasonable estimate of SP and the emulating NN complexity.

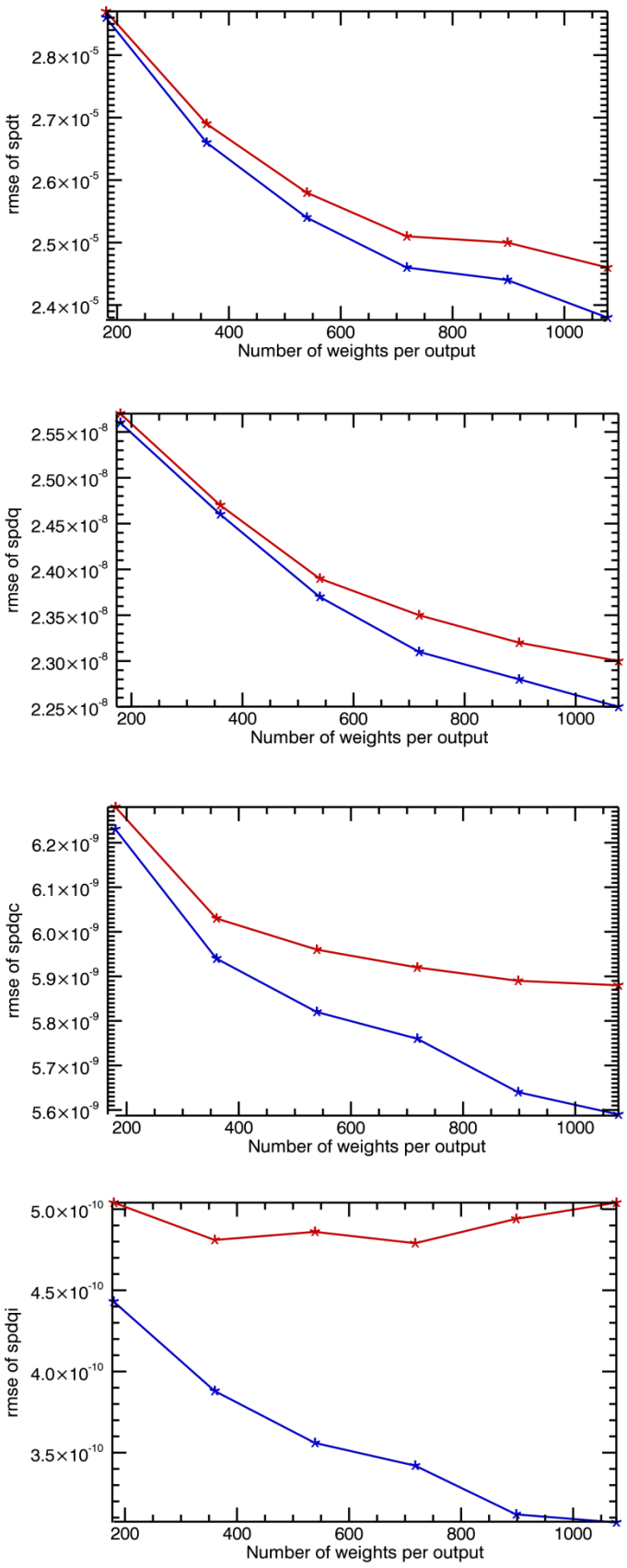

Fig. 3 Convergences of the approximation RMSEs for four prognostic outputs (profiles) of SP: *spdt*, *spdq*, *spdqc*, and *spdqi* are shown. Blue lines show RMSEs calculated using the training set and red – the independent test set. RMSEs are shown vs. the NN complexity per output, n_c. The figure shows RMSEs for NNs with k = 25, 50, 75, 100, 125, and 150 each (shown with asterisks). Corresponding values of n_c are shown in Table 7.

5.2 NN Ensemble

After all parameters of the emulating NN have been selected, an ensemble of ten NNs with n = 746 inputs, k = 50 hidden neurons, and m = 121 outputs normalized using (10) has been trained. The NN ensemble was generated using different initial values for NN weights to initialize the NN training. Ten different initial values (all small random numbers) for NN weights have been selected following the initialization algorithm developed by Nguyen and Widrow (1990).

The NN ensemble has been trained to emulate the mapping (5), because this mapping is a stochastic mapping (see discussions in 2.3.2 and in Section 6 below). A single NN may not be an adequate tool for emulating stochastic mappings, which may be better emulated using a NN ensemble (Krasnopolsky at al. 2011). This ensemble of ten NNs has been used in all following calculations as an emulation of SP. In the following sections it will be called NN-SP. The ensemble mean (EM) was calculated as a conservative mean, i.e., a simple statistical average of ensemble member outputs. Such an ensemble is called the conservative one. For example, an NN EM for *spdt* profile is calculated as,

$$SPDT_i = \frac{1}{s}\sum_{j-1}^{s} spdt_i^j \ , \ i = 1,\ldots,P \tag{13}$$

where P is the number of records in the validation data set, $spdt_i^j$ is a profile of *spdt* calculated by the NN ensemble member number j for the input data record (profile) number i, s (s = 10 in our case) is the number of NN ensemble members, and $SPDT_i$ is the EM (profile) of *spdt* for the data record number i.

6. Validation on an Independent Data Set

6.1 Calculated statistics

The NN ensemble described in Sect. 5.2 has been applied to the independent validation set described in Section 4 (Table 5). Various statistics have been calculated for each ensemble member NN, as well as for the conservative EM (13).

For each of ten ensemble member NN, for each of 121 outputs, nine statistics have been calculated. They include four outputs statistics (mean value, standard deviation, minimal and maximal value of the output variable) plus five error statistics (bias, RMSE, min error, max error, and correlation coefficient calculated vs. simulated SP data in the validation set). All-in-all, 1,089 statistics per ensemble member NN were calculated, plus 1,089 statistics for the EM, that is 11,979 statistics per the NN ensemble (10 members plus EM) totally. In addition, for 121 SP outputs available in validation datasets four output statistics (mean value, standard deviation, minimal and maximal value of the output variable) per output (484 statistics totally) were calculated.

To make sense of this enormous amount of information and reduce its volume, we combined together all outputs that represent a 3D variable in a profile and consider the profile statistics (integrated vertically over L vertical levels). The NN outputs were treated in the same way. Thus, we have five 3D variables (*spdt*, *spdq*, *spdqc*, *spdqi*, and *clouds*) and two 2D variables (*prect2d precst2d*). In this way, we obtained nine statistics per 3D or 2D output. They include four outputs statistics per se (mean value, standard deviation, minimal and maximal value of the output 3D or 2D variable) plus five error statistics (bias, RMSE, min error, max error, and correlation coefficient) calculated vs. simulated SP data in the validation set. As a result we

reduced 11,979 statistics to 630 ones for the NN ensemble totally. It is still a formidable task to analyze these 630 statistics.

6.2 Taylor diagrams for stochastic and deterministic outputs

To present 350 of 630 statistics in a vivid easily comprehensible way, we used the Taylor diagram (Taylor 2001). The Taylor diagram allowed us to present results for all seven SP outputs and all ten ensemble members together with the ensemble average in a single figure (Fig. 4). The diagram is plotted in polar coordinates (ρ,φ). Each point at the diagram corresponds to one of the members of the NN ensemble or to NN EM for one of seven SP output variables marked by one of seven symbols (see the legend of Fig. 4). For each of these points the radial distance from the origin to the point, ρ, is equal to the ratio of the NN output field standard deviation, σ_o, to the corresponding validation SP output field (from the validation set) standard deviation, σ_v,

$$\rho = \frac{\sigma_o}{\sigma_v} \tag{14}$$

The azimuthal angle $\varphi = \cos^{-1}(CC)$, where CC is the correlation coefficient between the NN output and the corresponding variable in the validation set. The azimuthal position of the point gives the CC between the two fields, and the radial lines are labeled by the values of CC at the outer circle. For all output variables from the validation data set $\rho = 1$ and CC = 1 ($\varphi = \cos^{-1}(1) = 0$). Thus, all outputs have the same reference point (1,0) at the horizontal axis of the diagram. The closer the output point to the reference point, the smaller the approximation error is. The dashed lines measure the distance from the reference point and, as a consequence indicate the centered RMS errors (or error standard deviations) normalized to σ_v. Thus, the radial position of the diagram point indicates the accuracy of NN in representing the output variance (the closer to the circle with the radius $\rho = 1$ the better), whereas the azimuthal angle reflects the phasing of the

NN output profiles with respect to validation profiles (the closer to the horizontal line the better). In addition to this information, the Taylor diagram shows a complimentary score, *S*, which characterizes NN skills in emulating the corresponding mapping. This score is calculated as,

$$S = \frac{4\cdot(1+R)}{(\rho+\frac{1}{\rho})^2\cdot(1+R_0)} \tag{15}$$

where R is the correlation coefficient CC, $R_0 = 1$ in our case and ρ is defined by (14). Isolines of *S* are shown by thin solid curves in the diagram.

Fig. 4 shows the Taylor diagram for all seven SP outputs and all ten ensemble member NNs together with the ensemble average. Different outputs and their EMs are presented with different symbols of different colors (see the figure legend); thus, a group of eleven points depicted with the same symbol and color presents results for a particular output of all ten NN ensemble members plus the EM for this output. Fig. 4 clearly suggests that SP has two qualitatively different types of outputs. For two outputs *spdqi* and *spdqc* NN results demonstrate much lower correlation with the validation data, when compared grid point by grid point, than NN results for *spdt*, *spdq*, *clouds*, and *prect2d*. The output *precst2d* is somewhere in between but closer to the second type.

Another characteristic feature of outputs of the first type and *precst2d* is that the ensemble spread for these outputs are significantly larger than that for the second type of the outputs. This significant spread indicates the higher level of uncertainty and stochasticity in the outputs of the first type; these outputs are "stochastic" outputs. The outputs of the second type we will call "deterministic" outputs.

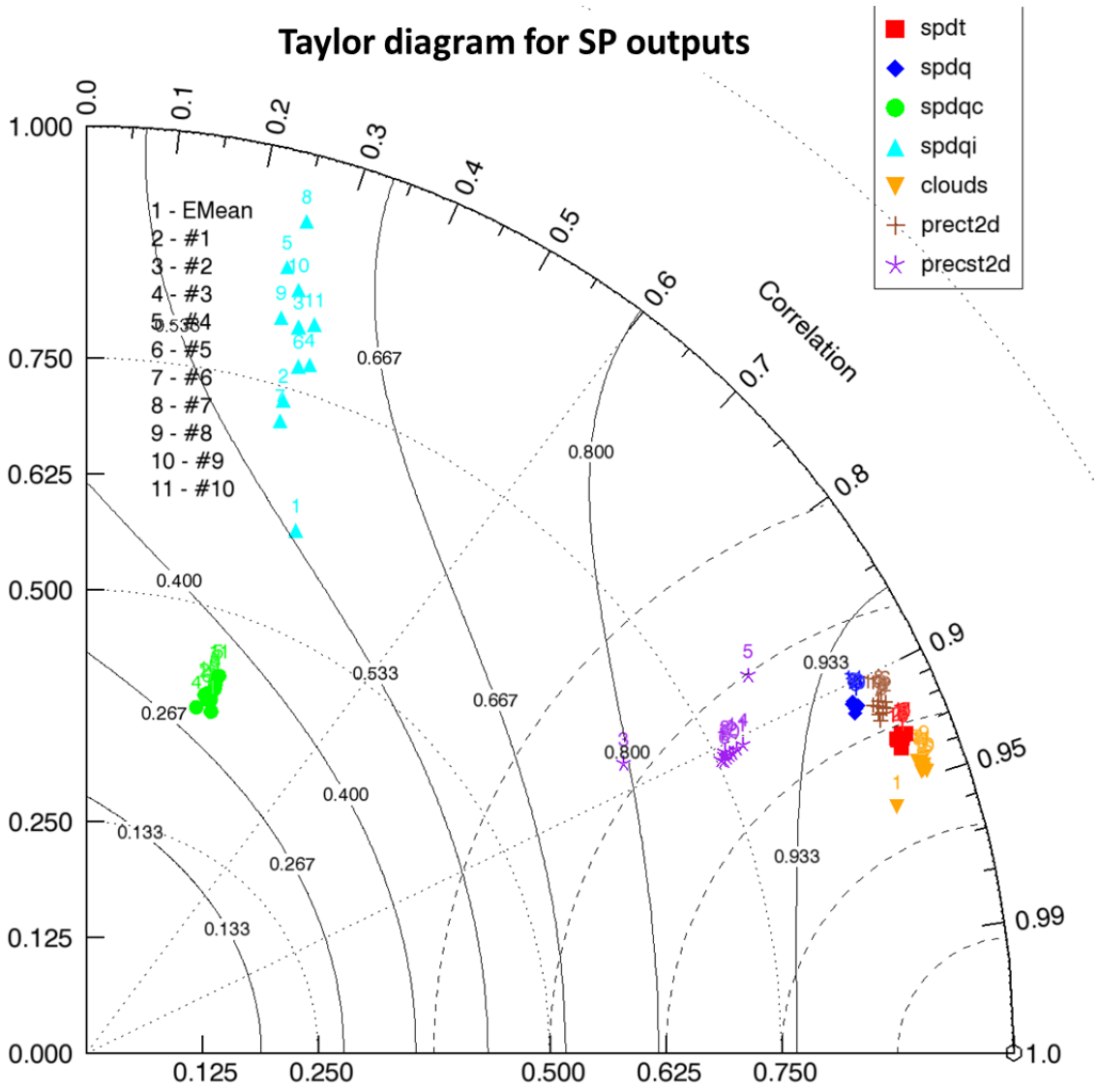

Fig. 4 Taylor diagram for all seven SP outputs of all ten ensemble member NNs together with the EM. Seven different outputs and their EMs are presented with different symbols of different colors (see the figure legend). Solid isolines identify skill score values, dashed lines indicate correlation. The number of the ensemble member indicated by a number written beside the symbol.

It is noteworthy that for the stochastic outputs the spread of the ensemble members is mainly in the radial direction. They do not show significant variations in correlation coefficient and RMSEs. Keeping these results in mind, let us look again at the conversions of NN outputs to SP outputs on the test set with the increase of the number of hidden neurons (NN complexity), which we investigated in Section 5.1.4 (Fig. 3). In that section we investigated the conversions

in terms of approximation RMSE; now, however, we will use a Taylor diagram to observe the conversions from a different perspective.

Fig. 5 shows a Taylor diagram for all seven SP outputs of emulating NNs with six different numbers of hidden neurons (see the upper left legend). Seven outputs are presented with different symbols of different colors (see the upper right legend). There are no ensembles here; for each number of hidden neurons only one emulating NN is presented. As the previous figure, this figure demonstrates significant differences in behavior of stochastic and deterministic outputs when the number of hidden neurons (NN complexity) increases. The deterministic outputs converge to the reference point (1,0) on the Taylor diagram (perfect correlation, and variance). For these outputs RMSE decreases and CC increases; however, improvement in ρ is minor. The stochastic outputs behavior is very different; they move in the radial direction toward the circles with increasing ρ –> 1. It means that the variability (standard deviation σ_o) of this NN output converges to variability of the corresponding SP output. For *spdqi*, ρ almost reaches the unit circle; for *spdqc* it steadily increases and reaches 0.5. However, the phasing (i.e., NN output profile to SP output profile correspondence) does not improve significantly; that is RMSE and CC do not improve significantly. It is noteworthy, that there is also a noticeable improvement in emulating NNs performances for stochastic outputs in terms of the score *S*, because this score takes into account ρ as well.

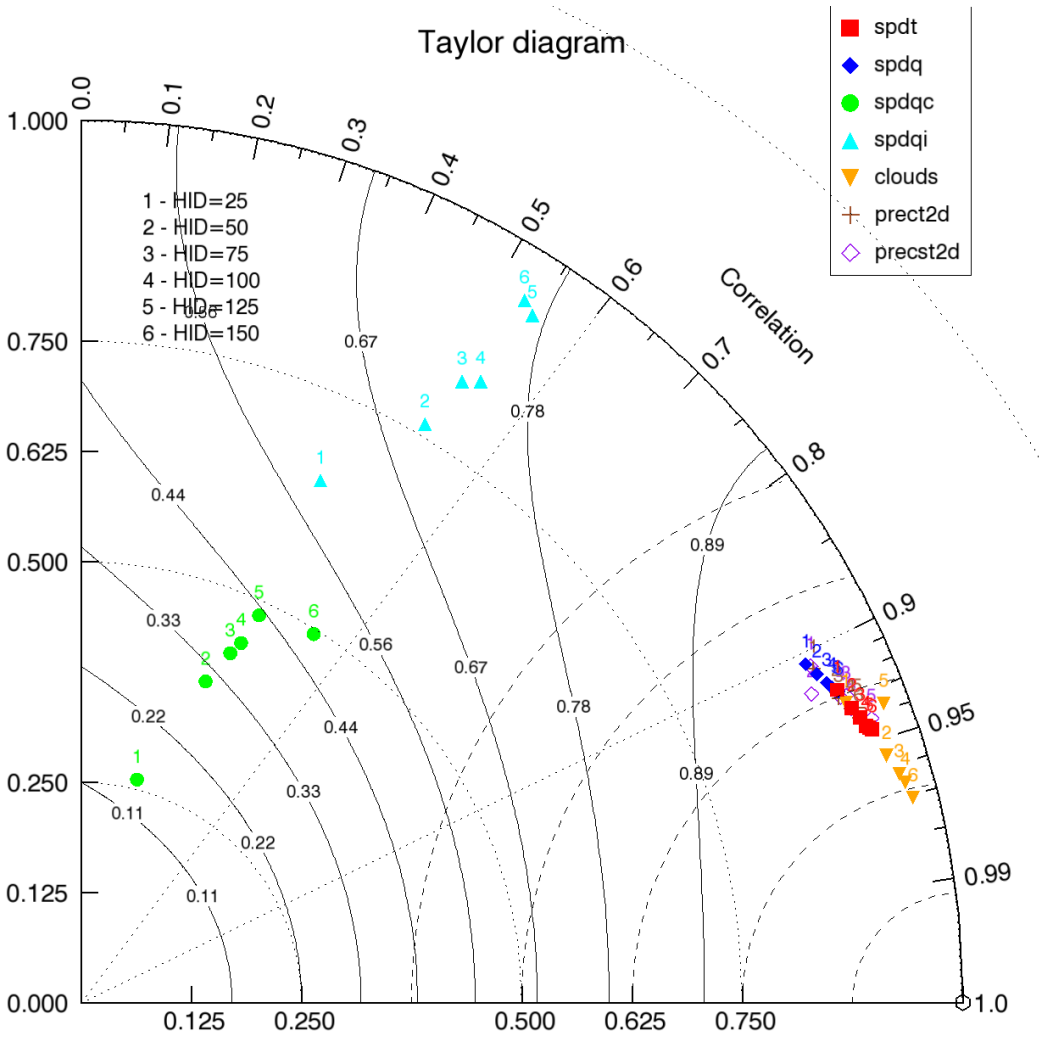

Fig. 5 Taylor diagram for all seven SP outputs of emulating NNs with six different numbers of hidden neurons (see the upper left legend). Seven different outputs are presented with different symbols of different colors (see the upper right legend).

6.3 More approximation statistics

To obtain more detailed information about the approximation quality of the emulating NNs, several different statistics are plotted below. The error statistics are:

(i) RMSE (12);

(ii) RMSE profile calculated as,

$$RMSE_j = \sqrt{\frac{1}{P}\sum_{i=1}^{P}\left[Y_i^j - Y_{i\,NN}^j\right]^2}; \quad j = t+1,\ldots,t+L \tag{16}$$

where Y_i^j and $Y_{i\,NN}^j$ are outputs (without normalization) from the original SP and its NN emulation, respectively, the index $i=1,\ldots, P$ determines the record number in the validation data set, P is the number of the records, and $j = t+1,\ldots,t+L$ is the vertical index of the corresponding profile in the SP output vector (t is the index of the first profile component in the output vector Y), here L is the number of the profile vertical components;

(iii) σRMSE profile calculated as,

$$\sigma RMSE_j = \frac{RMSE_j}{\sigma_j}; \quad j = t+1,\ldots,t+L \tag{17}$$

where RMSE profile component $RMSE_j$ at each vertical level is divided by the standard deviation of the output at this level σ_j

(iv) Mean output profile, calculated over the entire validation data set of P records;

(v) Probability density function (pdf) for an output over the entire validation data set of P records.

6.3.1 3D deterministic outputs

First, we consider the results for three deterministic 3D outputs *spdt*, *spdq*, and *clouds*. Fig. 6 shows RMSE (16) and σRMSE (17) (left and right panels respectively) profiles for *spdt* for ten NN ensemble members (blue curves) and EM (thick red curve). The absolute error (RMSE) decreases with the height, while the relative error (σRMSE), normalized to the level variability σ, increases because variability decreases faster than the error. RMSE sharply increases when approaching the surface (L = 28), while σRMSE sharply decreases because, approaching the surface, the variability of the data increases much faster than the error. σRMSE never exceeds one sigma. The spread of NN ensemble members is small (as in Figs. 4 and 5) and the ensemble mean slightly reduces both errors.

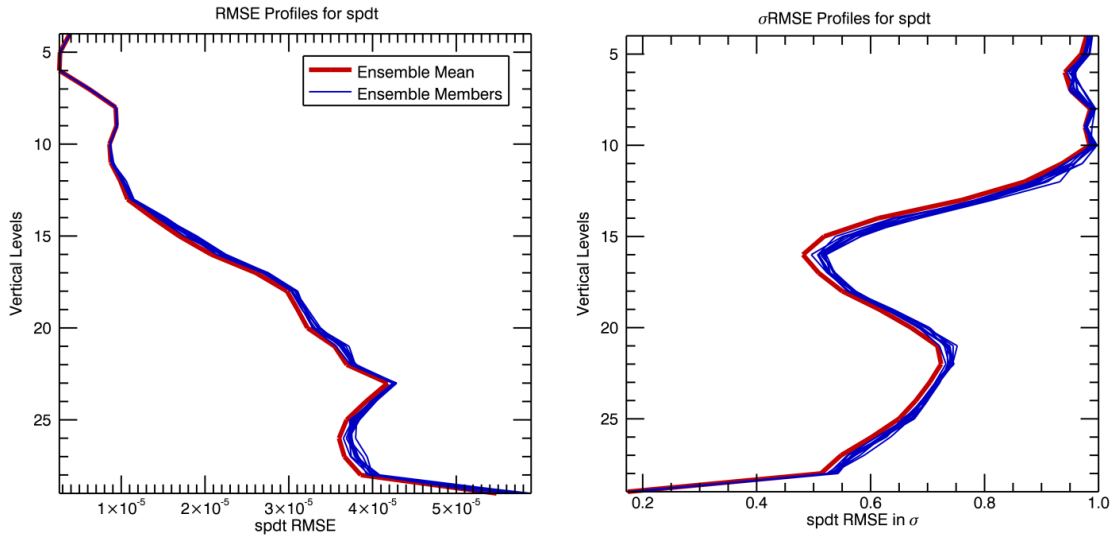

Fig. 6 RMSE (left panel) and σRMSE (right panel) profiles for *spdt* for ten NN ensemble members (blue curves) and EM (thick red curve).

Actually only testing NN-SP in the MMF can give the final answer to the question if the rate of decreasing the absolute error with height (Fig. 6, left) is sufficient for the accurate performance of NN-MMF. If it shows that the magnitudes of relative errors (Fig. 6, right) at higher model

levels are important, the normalization of outputs has to be changed to (9) or/and the higher weight have to be subscribed to the higher level errors in the error function during the training.

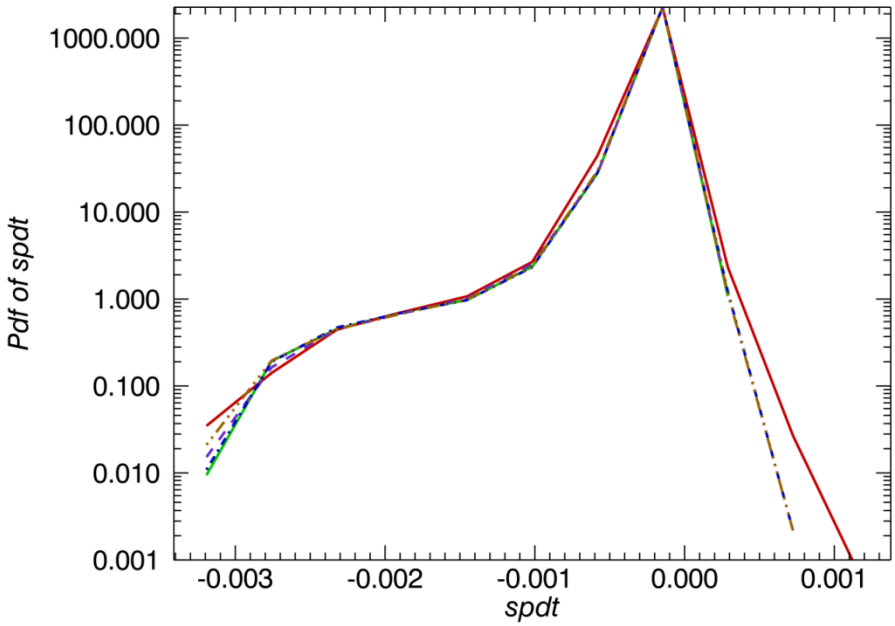

Fig. 7 Probability density functions for *spdt* for three NN ensemble members (dashed, dot-dashed and two dots-dashed curves), EM (green solid line) and SP output (red solid line). All vertical levels are included here and in Figs. 10, 13, 16 and 19. The vertical axis has a logarithmic scale.

Fig. 7 shows pdfs for *spdt* for three NN ensemble members, EM, and SP output. Ensemble members and EM reproduce pdf of SP output very well, slightly (notice the logarithmic scale of the vertical axis) reducing values at the far tails. The asymmetry of pdf is very well reproduced.

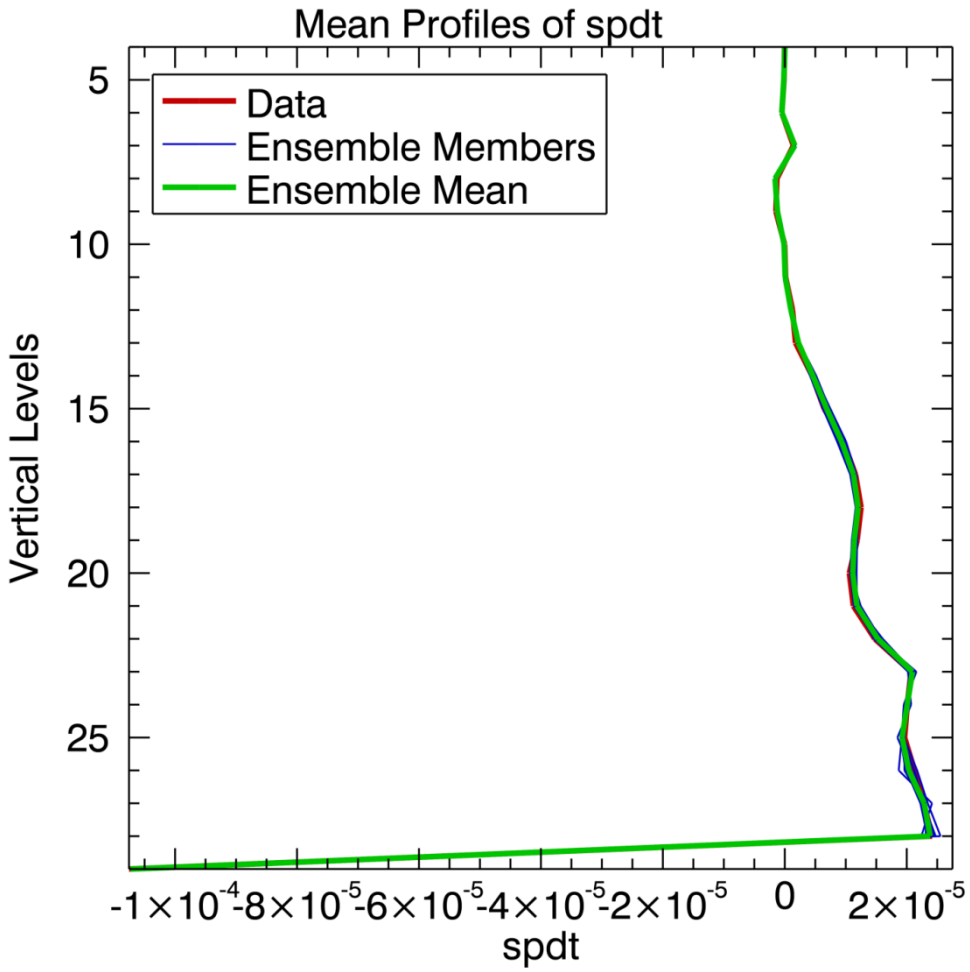

Fig. 8 Mean vertical profile of *spdt* for ten NN ensemble members (blue curves), EM (thick green curve), and the SP output (thick red curve).

Fig. 8 presents mean vertical profiles of *spdt* calculated over the entire validation data set using ten NN ensemble members (blue curves), EM (thick green curve), and the SP output (thick red curve). All NN ensemble members as well as EM represent the validation profile sufficiently well.

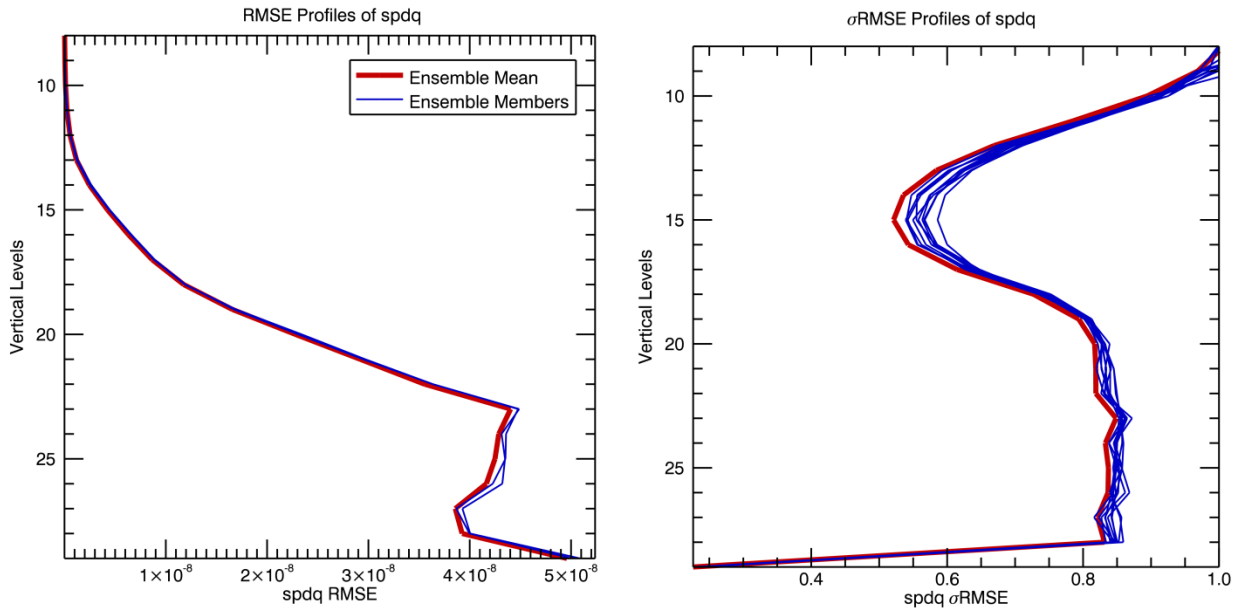

Fig. 9 RMSE (left panel) and σRMSE (right panel) profiles for *spdq* for ten NN ensemble members (blue curves) and EM (thick red curve).

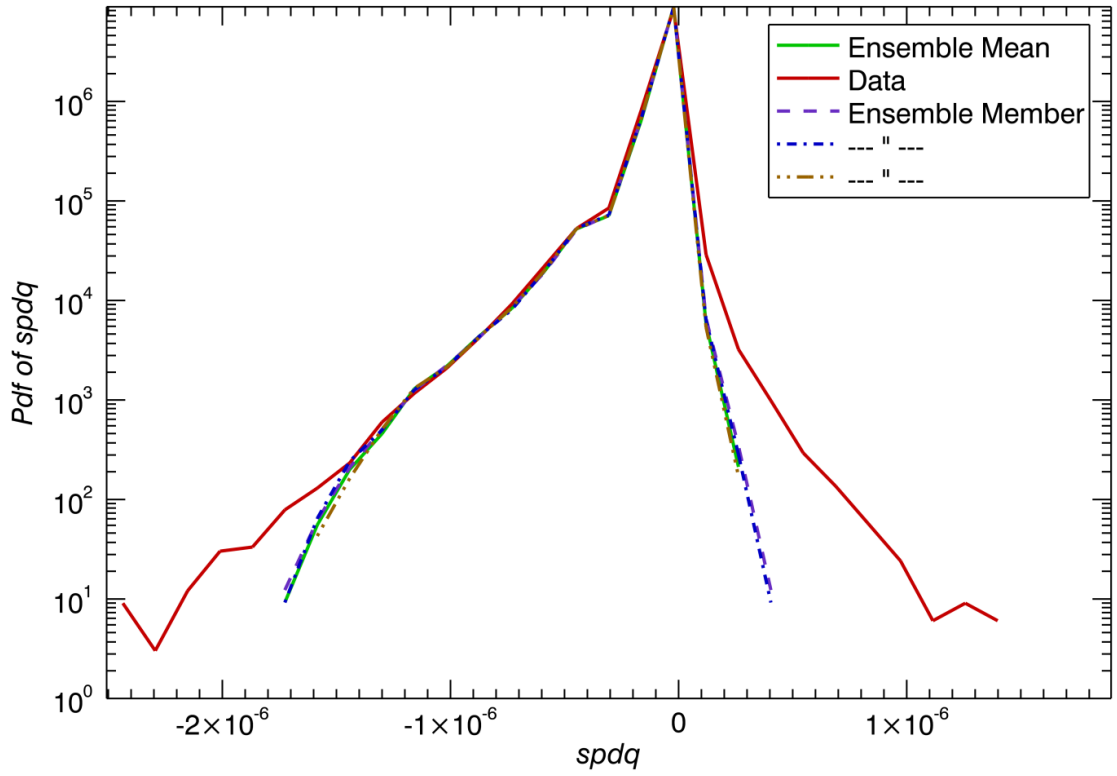

Fig. 10 Probability density functions for *spdq* for three NN ensemble members (dashed, dot-dashed and two dots-dashed curves), EM (green solid line) and SP output (red solid line). The vertical axis has a logarithmic scale.

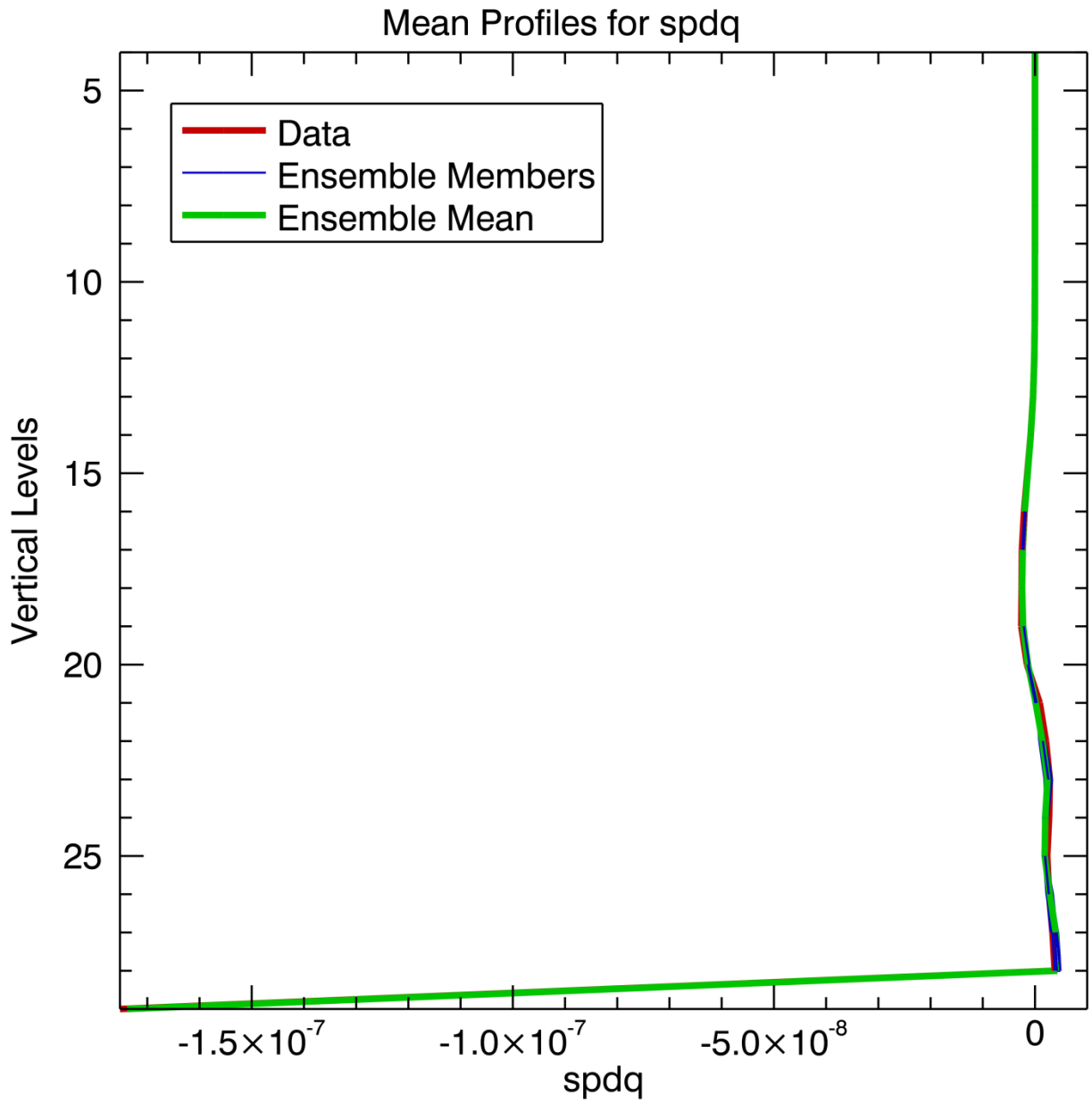

Fig. 11 Mean vertical profile of *spdq* for ten NN ensemble members (blue curves), EM (thick green curve), and the SP output (thick red curve).

Fig. 9 shows RMSE (16) and σRMSE (17) (left and right panels respectively) profiles for *spdq* for ten NN ensemble members (blue curves) and EM (thick red curve). The behavior of errors is similar to that for *spdt* (Fig. 6). Here also σRMSE never exceeds one sigma. The spread of NN ensemble members is also small (as in Figs. 4 and 5) and ensemble mean slightly reduces both errors. Fig. 10 shows pdfs for *spdq* for three NN ensemble members, EM, and SP output.

Ensemble members and EM reproduce pdf of SP output rather well; they reproduce even the asymmetry of the curve. However, both tails of pdf are suppressed in NN's pdfs, which leads to a slight reduction of the variability of NN generated *spdq*, which can be seen in Figs. 4 and 5. The reduction of the variability is minimal because the tails of pdf have very small magnitude (notice the logarithmic scale of the vertical axis). Fig. 11 presents mean vertical profiles of *spdq* calculated over the entire validation data set using ten NN ensemble members (blue curves), EM (thick green curve), and the SP output (thick red curve). All NN ensemble members as well as EM represent the validation profile sufficiently well.

Figs. 12 to 14 present the same statistics for *clouds* SP output. In this case the spread of NN ensemble members for RMSE and σRMSE (Fig. 12) is greater than for two previous outputs and, therefore, the reduction of these errors by EM is more pronounced. As for pdfs (Fig. 13), NNs slightly increase the amount of low and medium cloudiness and slightly reduce the amount of heavy cloudiness as compared with the SP produced cloudiness. EM for heavy cloudiness (the green line) is closer to SP (the red line) than for all ensemble members. The mean *clouds* profile (Fig. 14) is represented by the members of NN ensemble and by EM sufficiently well, better than the temperature and water vapor.

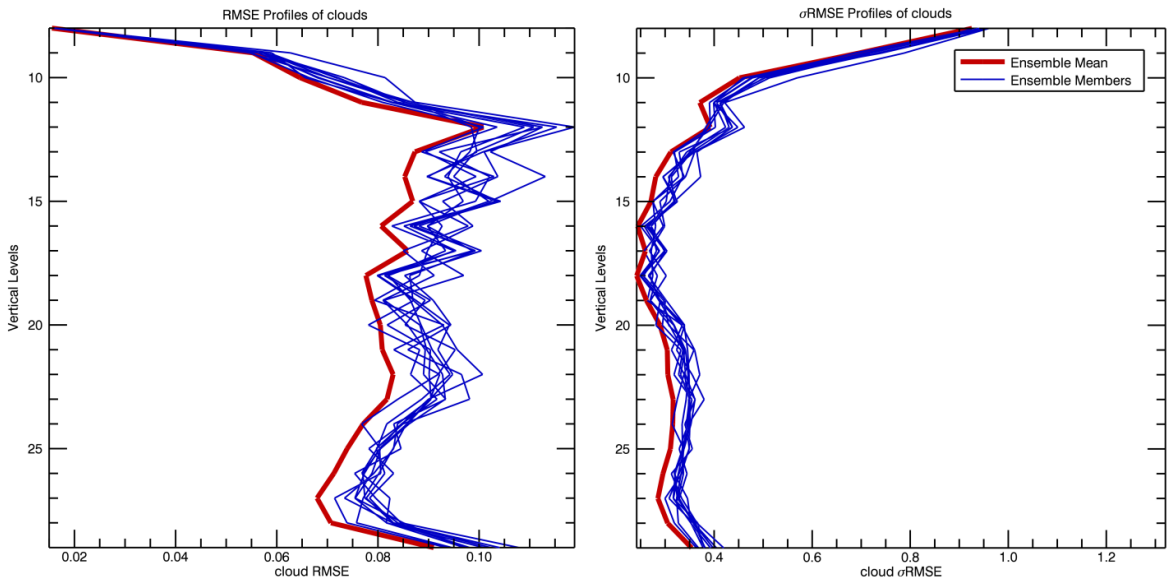

Fig. 12 RMSE (left panel) and σRMSE (right panel) profiles for *clouds* for ten NN ensemble members (blue curves) and EM (thick red curve).

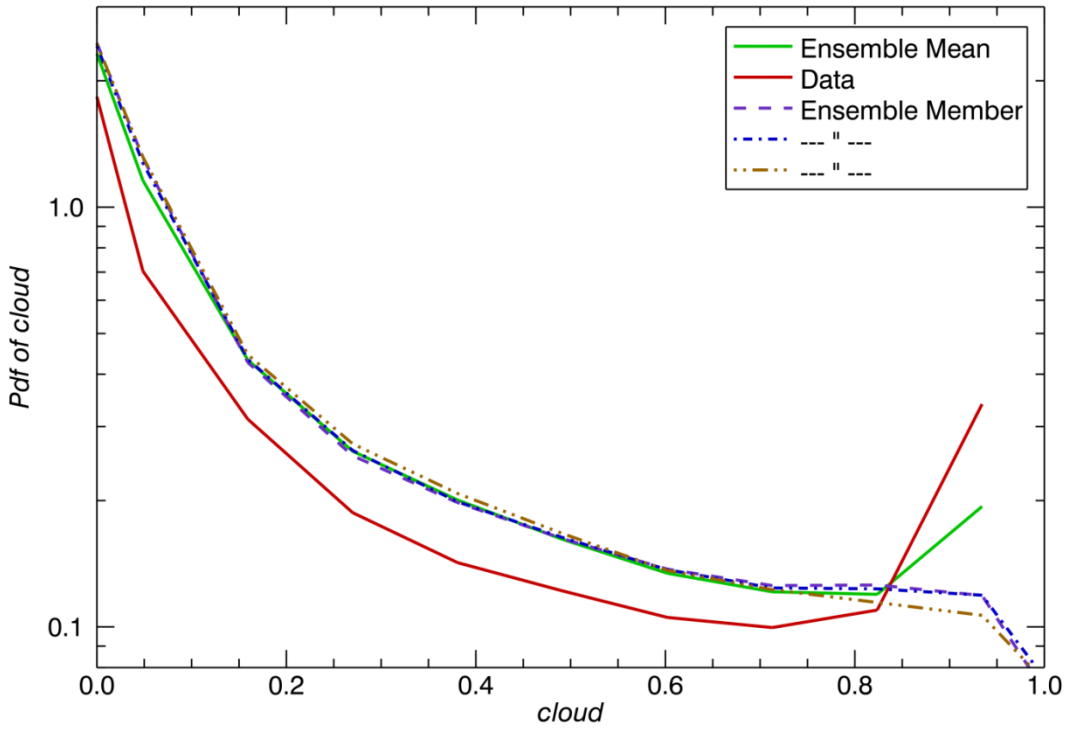

Fig. 13 Probability density functions for *clouds* for three NN ensemble members (dashed, dot-dashed and two dots-dashed curves), EM (green solid line) and SP output (red solid line). The vertical axis has a logarithmic scale.

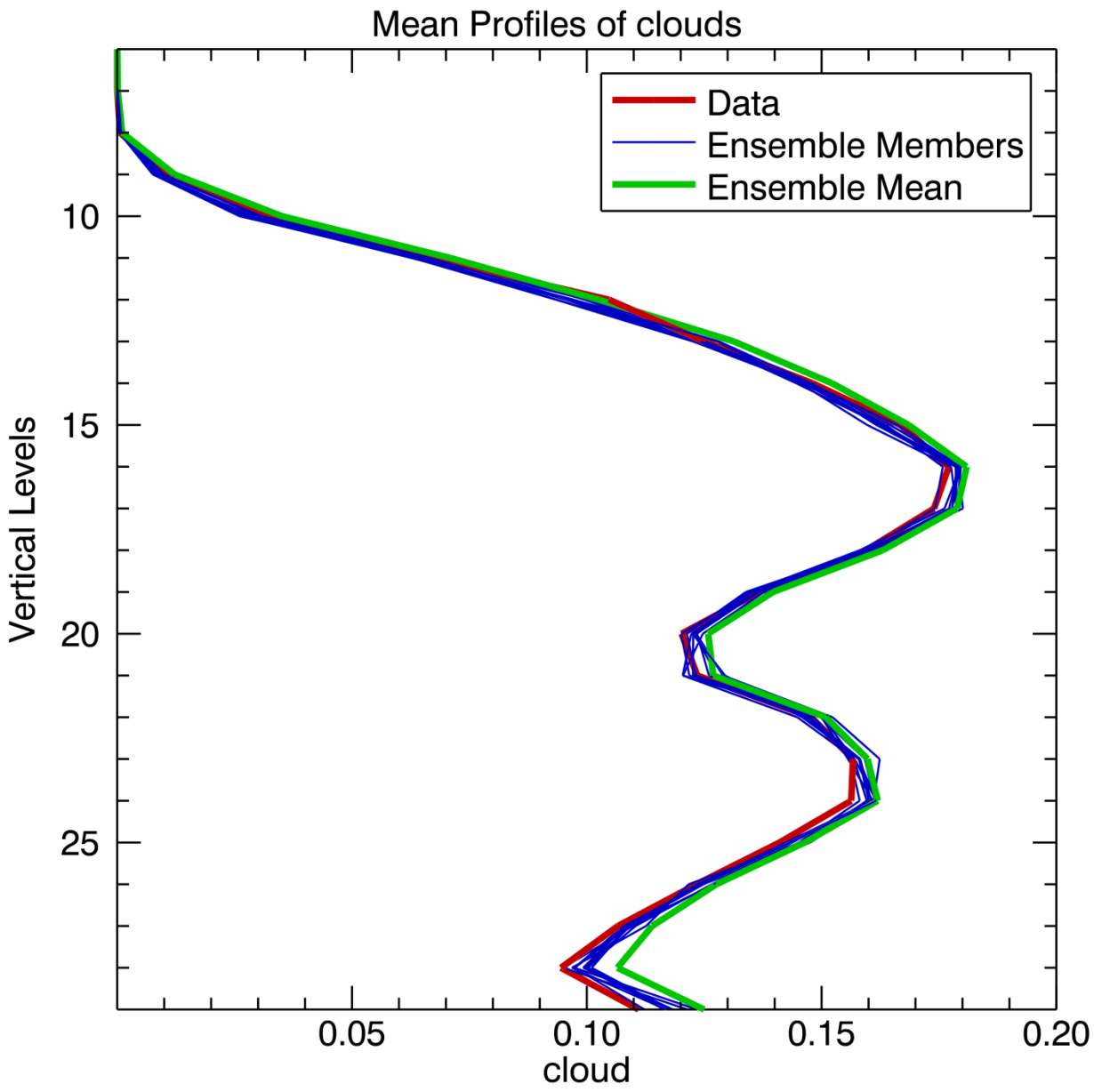

Fig. 14 Mean vertical profile of *clouds* for ten NN ensemble members (blue curves), EM (thick green curve), and the SP output (thick red curve).

6.3.2 3D stochastic outputs

Next we consider the results for two stochastic 3D outputs *spdqi* and *spdqc*.

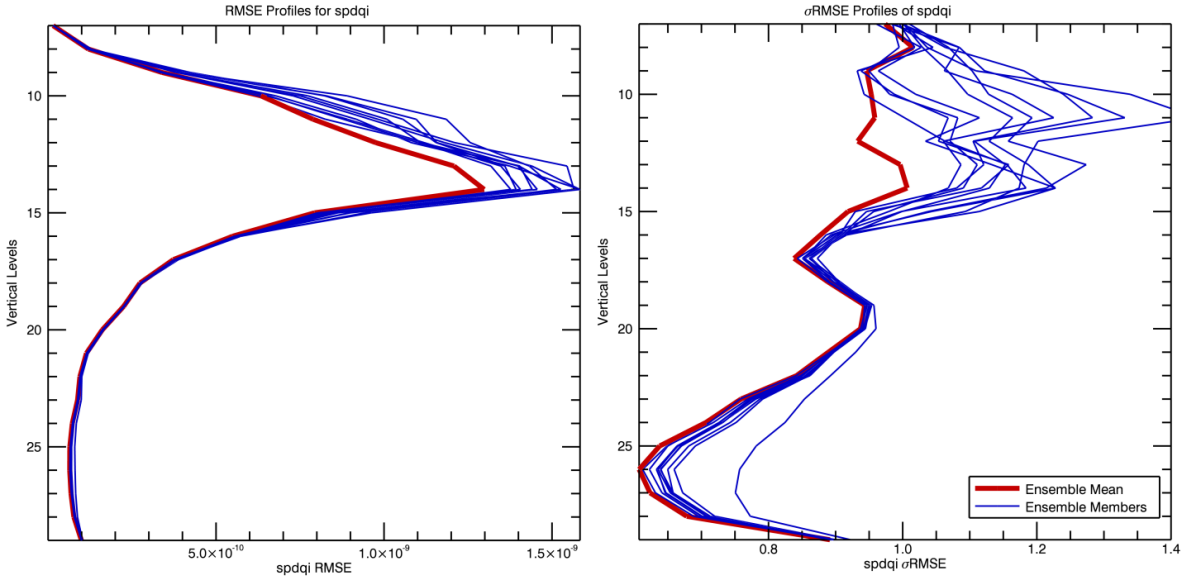

Fig. 15 RMSE (left panel) and σRMSE (right panel) profiles for *spdqi* for ten NN ensemble members (blue curves) and EM (thick red curve).

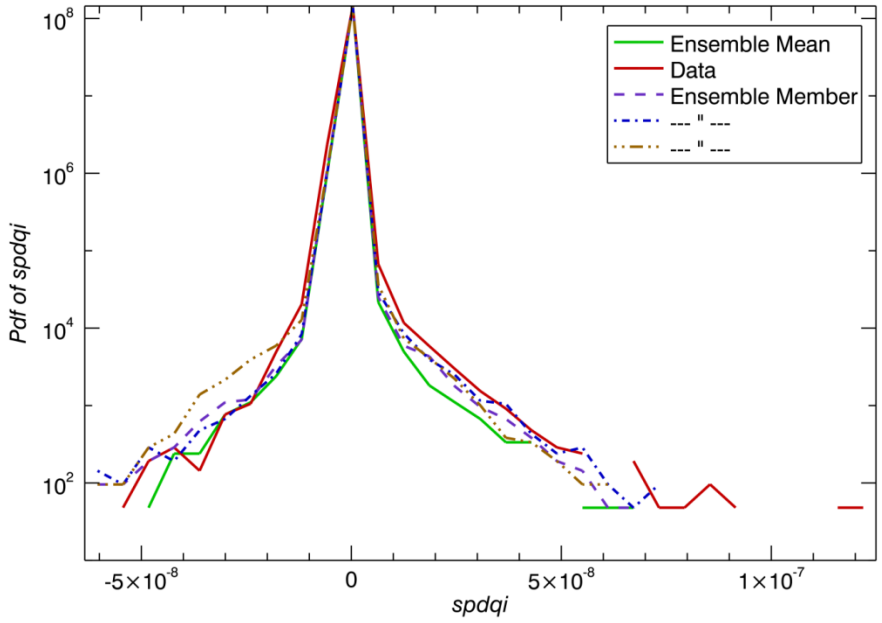

Fig. 16 Probability density functions for *spdqi* for three NN ensemble members (dashed, dot-dashed and two dots-dashed curves), EM (green solid line) and SP output (red solid line). The vertical axis has a logarithmic scale.

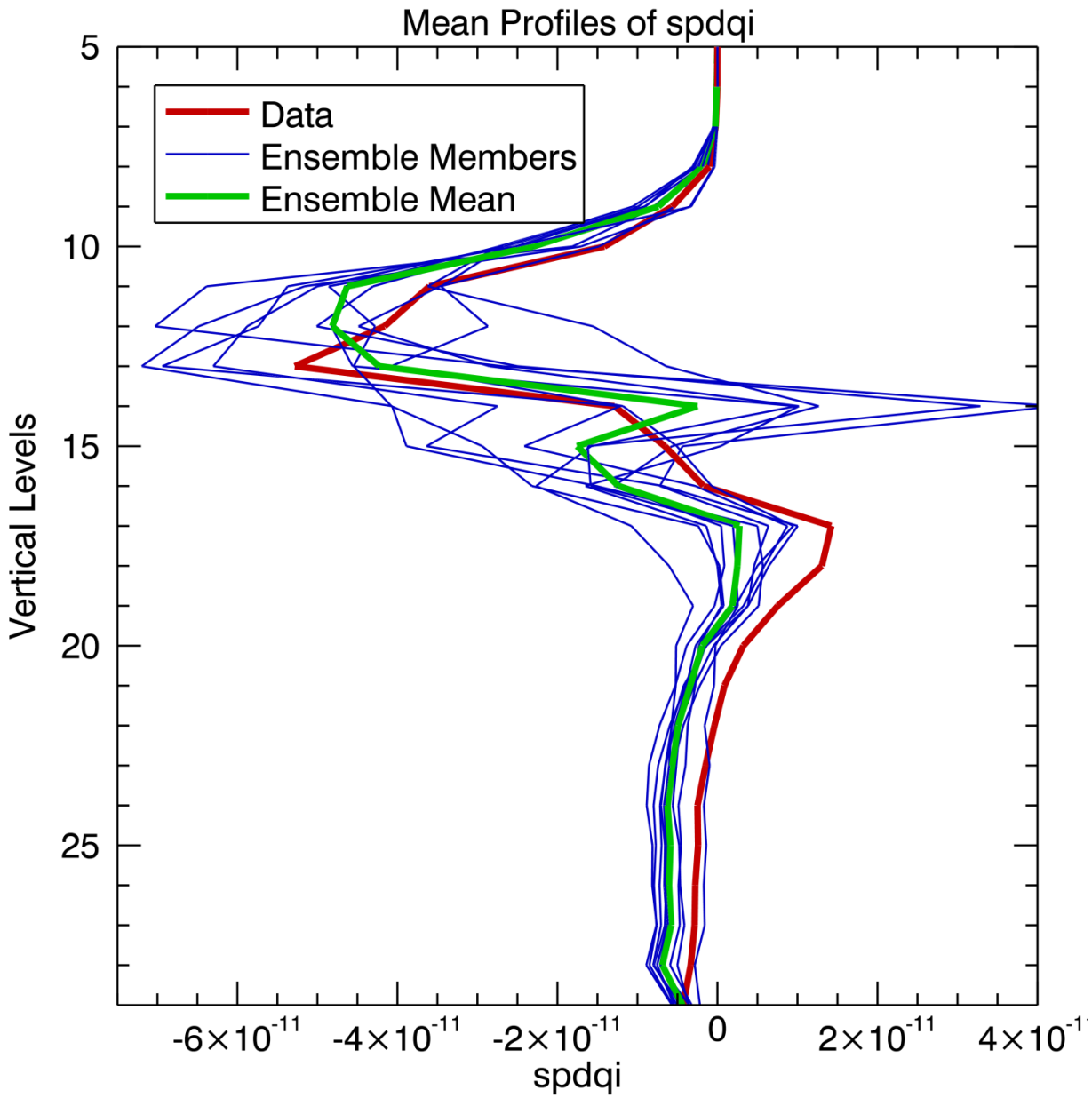

Fig. 17 Mean vertical profile of *spdqi* for ten NN ensemble members (blue curves), EM (thick green curve), and the SP output (thick red curve).

Fig. 15 shows RMSE (16) and σRMSE (17) (left and right panels respectively) profiles for *spdqi* for ten NN ensemble members (blue curves) and EM (thick red curve). The ensemble members demonstrate significant spread especially for higher levels, which indicates significant uncertainty in *spdqi.* σRMSE enhances the presentation of the *spdqi* uncertainty. The EM (red curve) reduces the uncertainty of *spdqi* significantly. For EM, σRMSE never exceeds one sigma.

Fig. 16 shows pdfs for *spdqi* for three NN ensemble members, EM, and SP output. Ensemble members and EM reproduce pdf of SP output rather well; they also reproduce a slight asymmetry of the distribution. Even both tails of pdf are represented satisfactory. Fig. 17 presents mean vertical profiles of *spdqi* calculated over the entire validation data set using ten NN ensemble members (blue curves), EM (thick green curve), and the SP output (thick red curve). NN ensemble members have a significant spread around the SP profile, while EM represents the validation profile rather well.

Fig. 18 shows RMSE (16) and σRMSE (17) (left and right panels respectively) profiles for *spdqc* for ten NN ensemble members (blue curves) and EM (thick red curve). The ensemble members demonstrate significant spread at σRMSE plot, which support conclusion that σRMSE is significantly more sensitive to the uncertainty in the data than RMSE. The EM (red curve) significantly reduces the uncertainty of *spdqc*. For EM, σRMSE never exceeds one sigma. Fig. 19 shows pdfs for *spdqc* for three NN ensemble members, EM, and SP output. Ensemble members and EM reproduce pdf of SP output well only in a limited area around the maximum. Both tails of the pdf are significantly suppressed. Fig. 20 presents mean vertical profiles of *spdqc* calculated over the entire validation data set using ten NN ensemble members (blue curves), EM (thick green curve), and the SP output (thick red curve). NN ensemble members have a significant spread around the SP profile; they strongly oscillate around the SP profile, while EM reduces the amplitude of these oscillations and is closer to SP profile.

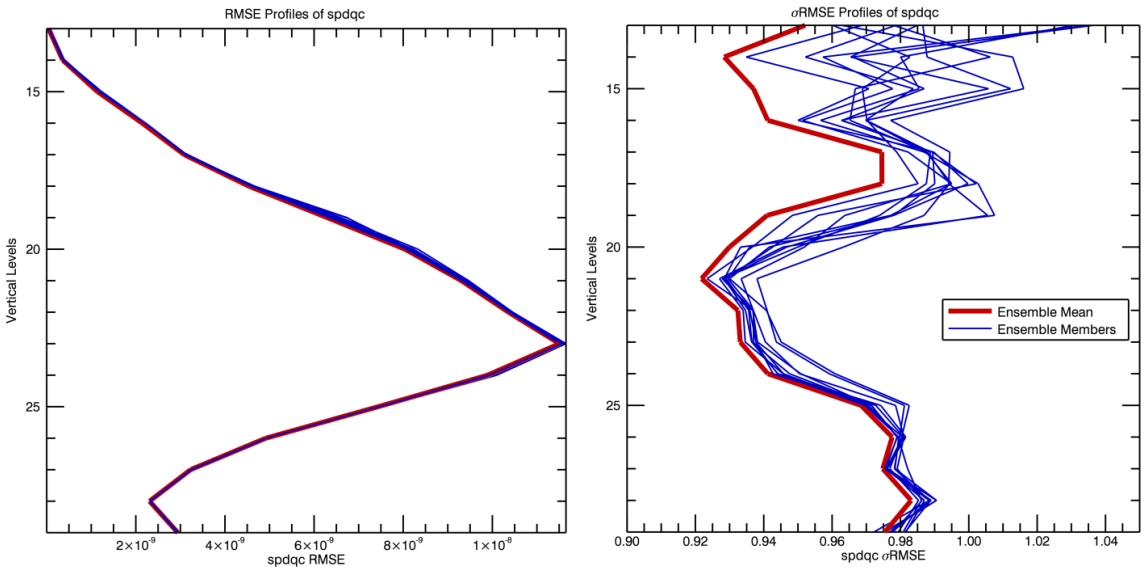

Fig. 18 RMSE (left panel) and σRMSE (right panel) profiles for *spdqc* for ten NN ensemble members (blue curves) and EM (thick red curve).

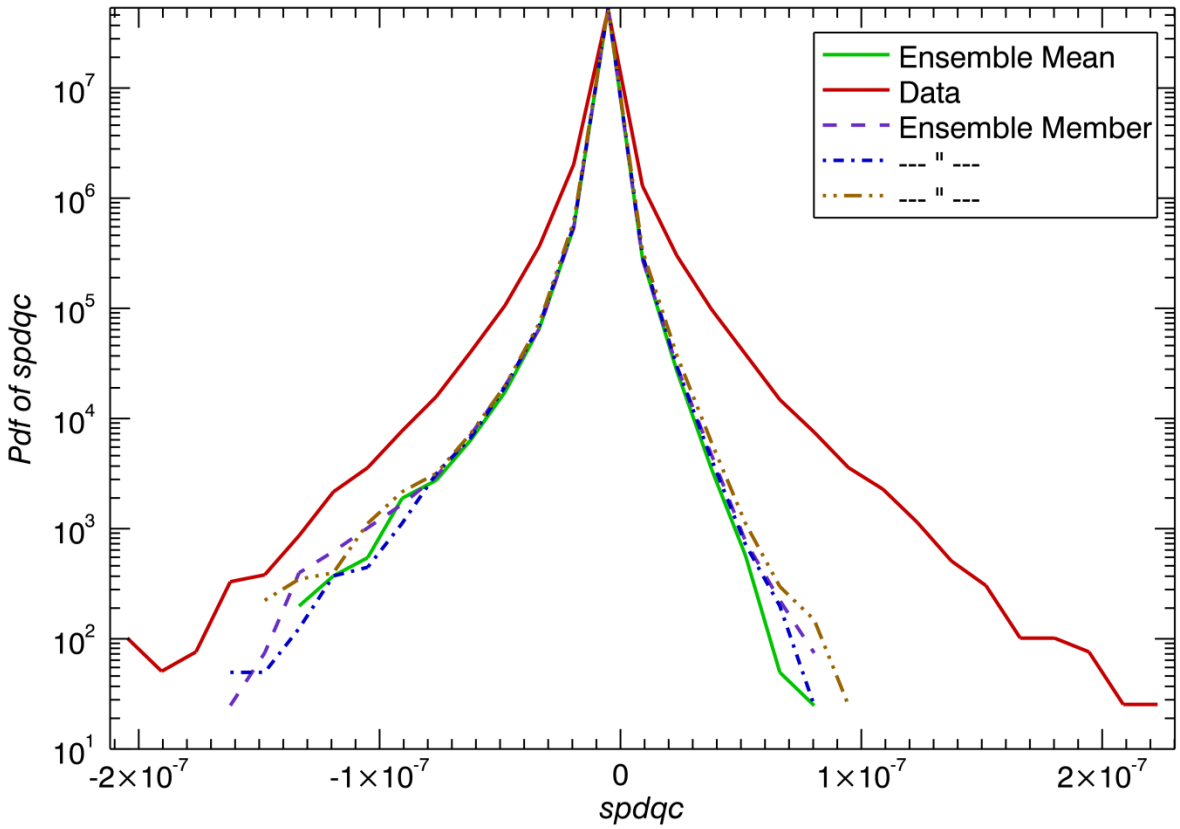

Fig. 19 Probability density functions for *spdqc* for three NN ensemble members (dashed, dot-dashed and two dots-dashed curves), EM (green solid line) and SP output (red solid line). The vertical axis has a logarithmic scale.

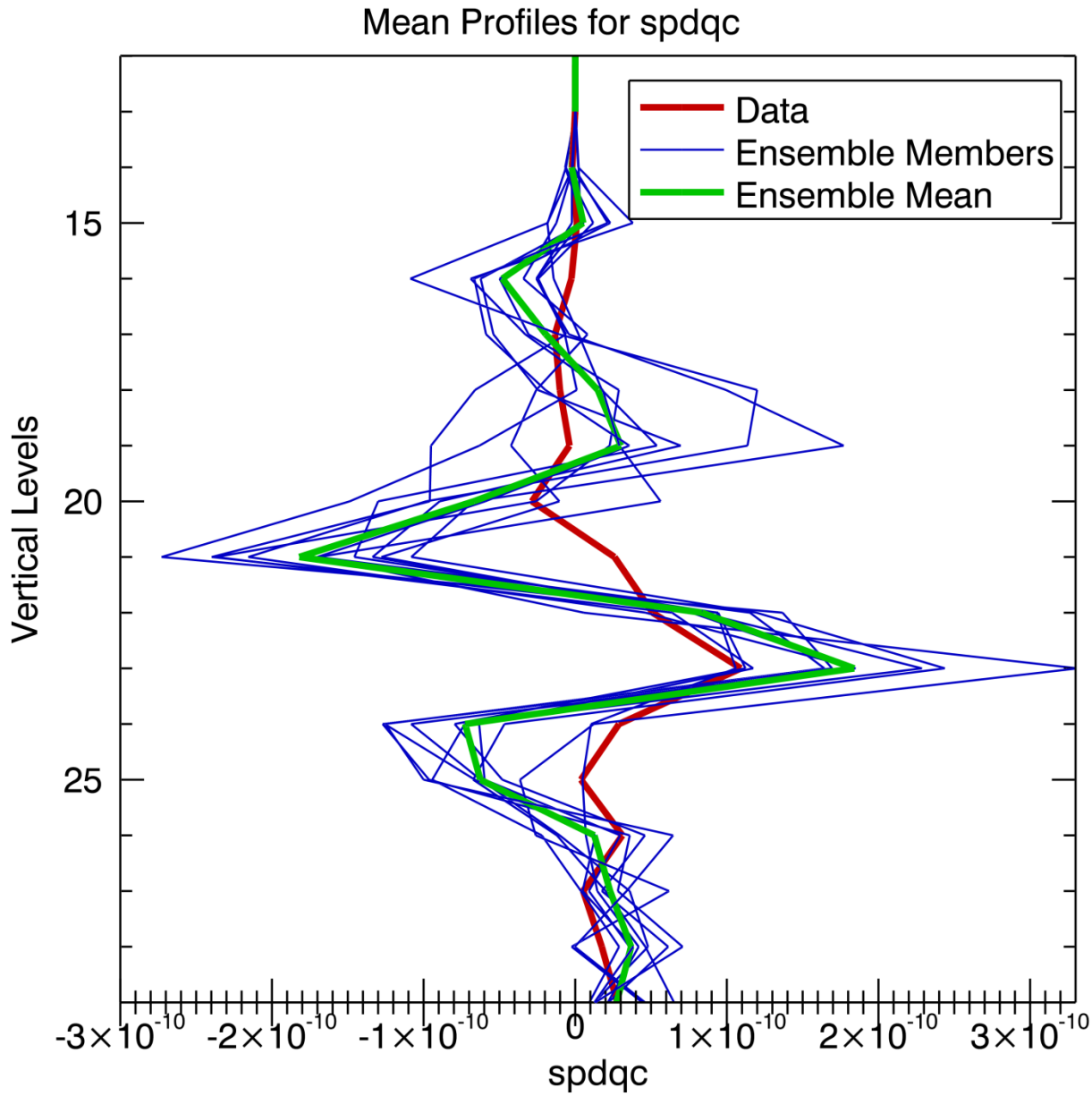

Fig. 20 Mean vertical profile of *spdqc* for ten NN ensemble members (blue curves), EM (thick green curve), and the SP output (thick red curve).

6.3.3 2D deterministic outputs

Finally, we consider results for two deterministic 2D outputs *prect2d* and *precst2d*. Fig. 21 show the standard deviation of NN approximation errors for *prect2d* vs. bias for *prect2d* for ten ensemble members (black asterisks) and EM (red diamond). The ensemble bias (a systematic error) is equal to the mean bias of the members as expected when using the simple conservative method to calculate the ensemble average. The EM error standard deviation (random error) is significantly smaller than the random errors of the ensemble members.

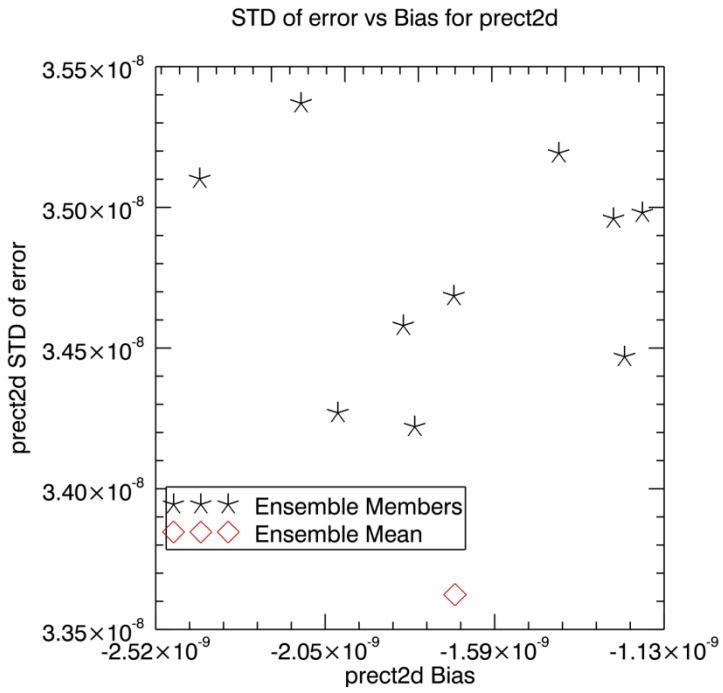

Fig. 21 Standard deviation of error vs. bias for *prect2d*. Ensemble members are shown with asterisk and EM with red diamond.

Fig. 22 shows the correlation coefficients for *prect2d* for ten ensemble members (black asterisks) and EM (red diamond). EM correlation coefficient is significantly better than those for NN ensemble members. Fig. 23 shows pdf for *prect2d*. Pdfs for three NN ensemble members (dashed, dot-dashed and two dots-dashed curves), EM (green solid line) and SP output (red solid line) are shown. The vertical axis has a logarithmic scale. EM represents the SP pdf very well.

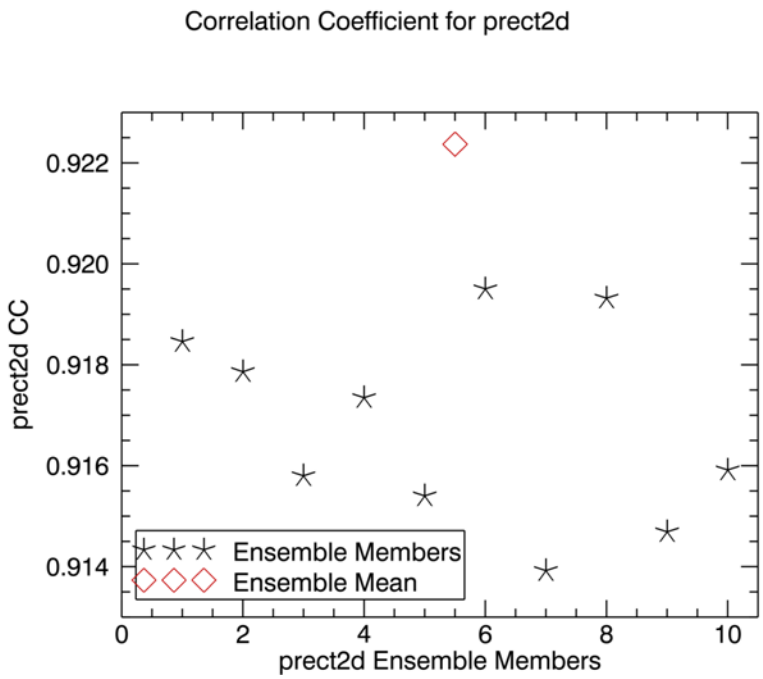

Fig. 22 Correlation coefficients for *prect2d* ensemble members are shown with asterisk and EM with red diamond.

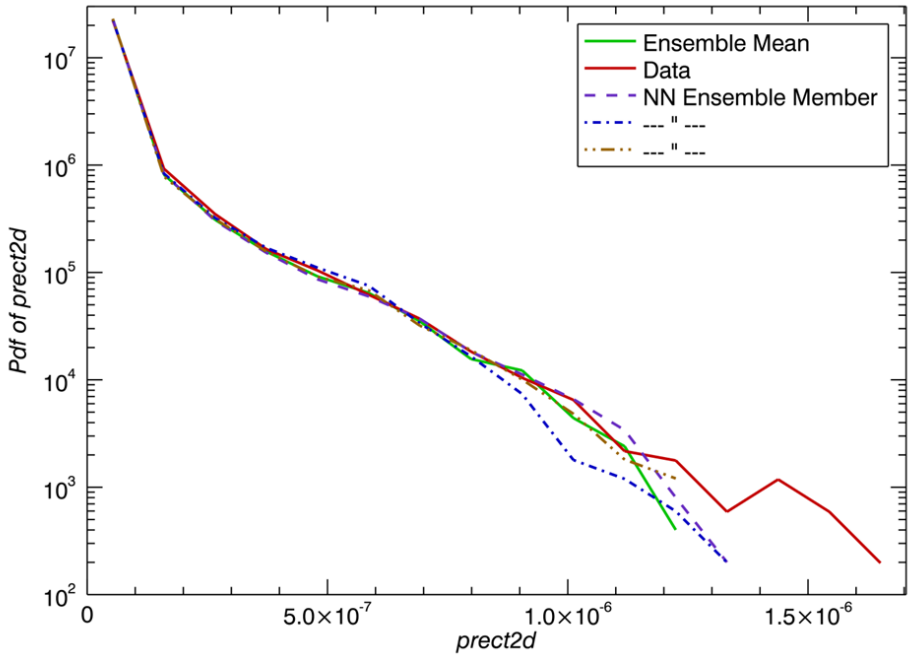

Fig. 23 Probability density functions for *prect2d* for three NN ensemble members (dashed, dot-dashed and two dots-dashed curves), EM (green solid line) and SP output (red solid line). The vertical axis has a logarithmic scale.

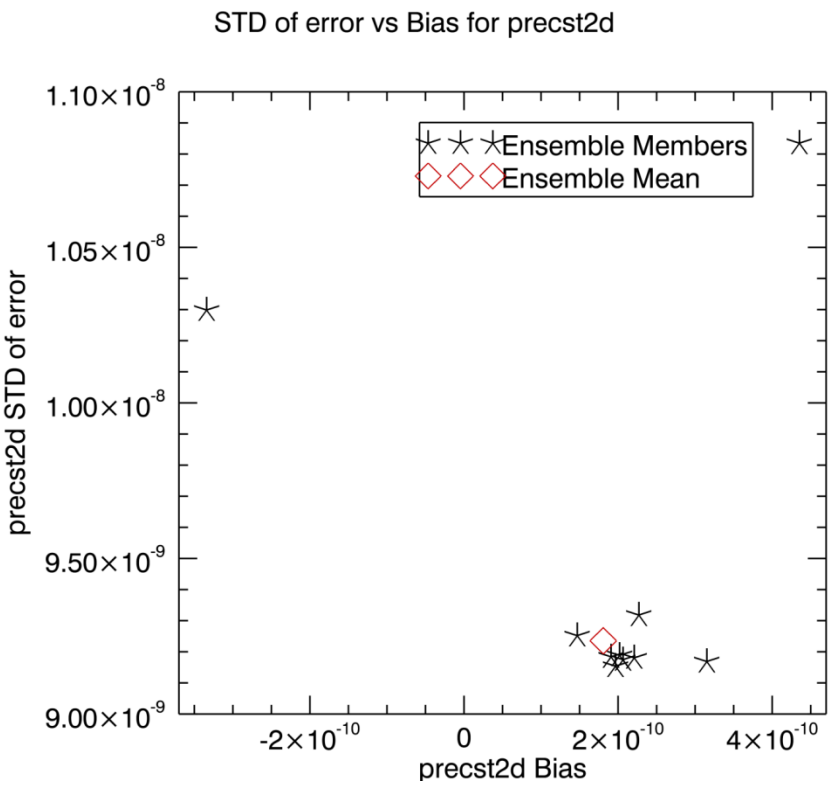

Fig. 24 Standard deviation of error vs. bias for *precst2d*. Ensemble members are shown with asterisk and EM with red diamond.

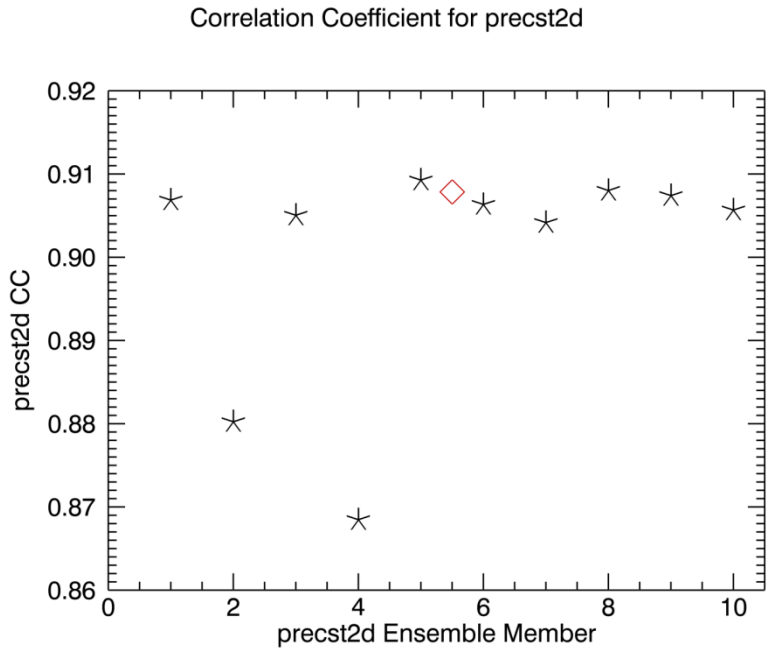

Fig. 25 Correlation coefficients for *precst2d e*nsemble members are shown with asterisk and EM with red diamond.

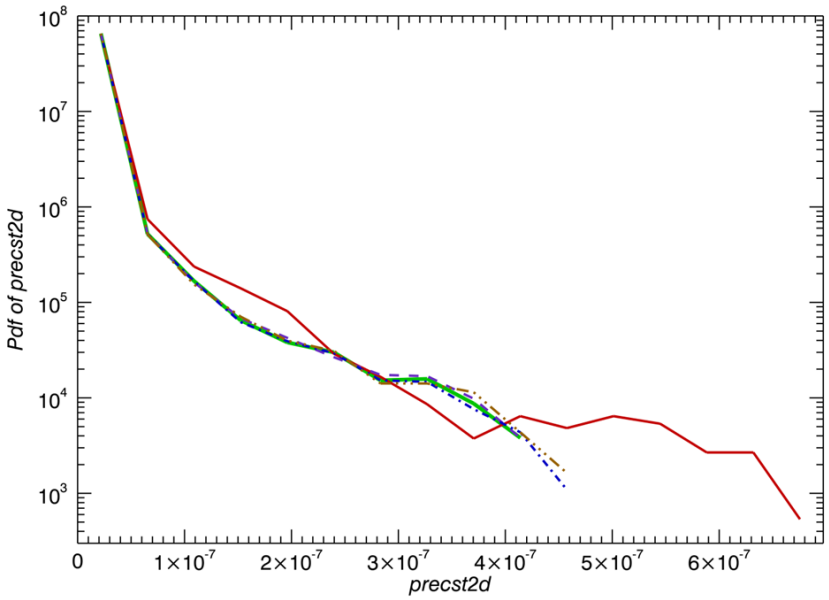

Fig. 26 Probability density functions for *precst2d* for three NN ensemble members (dashed, dot-dashed and two dots-dashed curves), EM (green solid line) and SP output (red solid line). The vertical axis has a logarithmic scale.

Figs. 24-26 show the same statistics for *precst2d* as Figs. 21-23 for *prect2d*. For *precst2d*, EM does not provide such very significant improvements as for *prect2d*; however, EM random error and CC are close to those of the best ensemble member. As for pdf, on average the EM pdf is close to the SP pdf.

6.4 Evaluation of the speedup for NN-SP

To estimate the calculation speedup, we used the MMF timing: one year of MMF simulations takes about 0.1million CPU hours. It provides an estimate 0.5s per SP call. For the emulating NN, 3,000 calls take approximately 0.6s, that is 0.0002s per call. Taking into account that one call of NN-SP consists of 10 NN calls (every time the ensemble of 10 NNs is evaluated), one NN-SP call takes approximately 0.002s. ***Thus, one NN-SP call is about 250 times faster than the SP call. It means that the NN-MMF will run at least as fast as the current CAM-5.*** It opens an attractive practical opportunity to extensively use and carefully validate the NN-MMF in a climate mode by running decadal and longer climate simulations.

6.5 Discussion

In this section we presented the results of validation of an ensemble of ten NNs emulating SP on an independent set of simulated data. The validation was performed on an independent data set, which was not used for NN training or for selecting the NN architecture. Despite the fact that the amount of data that we have for training and validation is limited (only four days of a MMF ran was saved), we believe that we can derive rather general conclusions from our study.

Two different types of SP outputs can be distinguished: two "stochastic" outputs *spdqi* and *spdqc* and five "deterministic" outputs. The stochastic outputs *spdqi* and *spdqc* effectively represent subgrid scale processes with smallest special and temporal scales that are much more sensitive to the uncertainty in the imbedded CRM initial conditions (see Section 2.3.2). This uncertainty results in the uncertainty in the corresponding SP outputs, which should be treated as stochastic outputs. The deterministic outputs are much better defined because they represent lager scale processes that are much less sensitive to the uncertain initial conditions of the imbedded CRM. The level of uncertainty of these outputs is much smaller than that of the stochastic ones, while it is still not zero. It was shown that the NN ensemble provides a more adequate tool for emulating SP:

1. For the deterministic SP outputs, the NN ensemble members and EM demonstrate a good level of approximation accuracy. They show low bias and RMSE and high CC with respect to the validation set, thus, demonstrating a good capability of reproducing deterministic outputs for each individual set of inputs. They also accurately reproduce

major statistics of deterministic outputs: variance, mean profile (for 3D outputs) or mean value (for 2D outputs), and pdf.

2. For the stochastic SP outputs, NN ensemble members demonstrate a reasonable capability of reproducing major statistics of stochastic outputs: variance, mean profile (for 3D outputs) or mean value (for 2D outputs) (and therefore, they have small bias), and pdf. EM significantly improves the results. However, the NN ensemble members and EM do not show a good level of approximation accuracy in reproducing outputs for each individual set of inputs (RMSE is not small and CC is not high enough).
3. Summarizing 2 and 3, for the deterministic outputs, the emulating NN provides good approximation both in terms of the regular approximation metrics (e.g., mean square error and CC) and in terms of data statistical properties (mean, variance, and pdf) as well. In other words, in this case NN approximates both statistical properties of the training set and the data set record by record. For the stochastic outputs *(*especially for *spdqi*), NN provides approximation in terms of statistical metrics (e.g., mean, variance, pdf). In this case it approximates the statistical properties of the training set; however, it does not provide a good enough approximation of the data record by record.
4. For the stochastic SP output *spdqc* the errors in approximating statistical properties of the output are higher (especially for the variance) than for the stochastic output *spdqi*. It may suggest that: (i) an important input information is missing, (ii) the length and representativeness of our limited data set is not sufficient for training this output, and/or (iii) this output is much more sensitive to the uncertainty in imbedded CRM initial conditions and some additional measures have to be taken to improve the NN accuracy for this output (see Conclusions).

The NN-SP provides an impressive, two orders of magnitude, speedup as compared with the original SP. It may provide a practical opportunity to use NN-MMF for decadal and longer climate simulations. It is noteworthy that all our estimates of the emulating NNs accuracy and speedup presented in this section are preliminary. The final conclusion about the quality of the emulating NN and its ability to substitute SP in the MMF runs can be derived only after validation of NN-SP in MMF, when parallel runs of the original MMF and NN-MMF are performed.

7. Conclusions

Some important methodological issues of the representation of a cloud "superparameterization" (SP) with a Neural Net emulation (NN) were discussed. The methodological issues include:

1. Certain fields used within the parameterization that modulate the response of SP output to input fields are not available to the NN. The uncertainty introduced by this has been interpreted as stochasticity; thus, the SP has been interpreted as a stochastic mapping and a NN ensemble has been selected as an adequate strategy for sampling from that stochastic ensemble.
2. The grid-box averaged fields normally used by the large scale model are used as inputs to the NN. An input vector *X* is defined containing these fields, and the aggregated results of internal components of the SP that are used to update the large scale model are identified as the output vector *Y*. A limited training dataset was drawn selectively from four days of MMF global simulations, spanning diurnal and latitudinal variability. It is our initial hypothesis that a single NN-SP can be constructed for all global conditions.
3. Experiments with different NN architectures were performed and assessed to optimize its ability to reproduce the SP module output fields, minimizing cost, and maximizing accuracy. These experiments include comparisons of a single emulating NN with many outputs vs. a battery of simpler emulating NNs with different outputs. Also, experiments with different output normalizations and different emulating NN complexities were conducted.
4. An ensemble of NN's were developed using a training dataset to identify the NN ability to reproduce the SP output and to characterize uncertainty associated with the stochastic aspect of the NN parameterization.

5. The ensemble of emulating NN's were evaluated using an independent (testing) dataset. Performance (accuracy and speedup) of the NN-SP was explored. Multiple statistics were used to evaluate NN-SP and the differences between SP and NN-SP.

The major results obtained in this work, which is a proof of a concept, allow us to expect that:

1. The SP can be emulated by NN with a satisfactory accuracy (based on validation on an independent data set);
2. The NN-SP provides an impressive, two orders of magnitude, speedup as compared with the original SP.

Based on promising initial results obtained in this work and discussed in Section. 6.5, our short-term plans include:

1. Perform parallel runs of an MMF with the original SP and an NN-MMF with the NN-SP developed in this study, and evaluate the accuracy and the performance of NN-SP through assessing the differences between the MMF and the NN-MMF simulations and computation times.
2. Extend the PNNL MMF runs to one or more years to create an extended and more representative data set that includes also seasonal and annual variability for training the NN-SP. If the estimated speedup will hold in the NN-MMF run, long global climate simulations with the NN-MMF will not be more time consuming than those using CAM-5.
3. If extending the training set will not be sufficient for reducing the differences between the MMF and the NN-MMF simulations to desired small values, an extended NN architecture, which could alleviate the memory/history problem of the approach, will be introduced.

An extended NN architecture can be introduced in the following way. Let us modify our NN-SP (6), modifying and expanding input vector by including in the modified input vector Z the values of X at one or several previous time steps,

$$Y_t = Net(Z_t), \tag{18}$$

where t is the current MMF time step and $Z_t = \{X_t, X_{t-1}, \ldots, X_{t-n}, Y_{t-1}, \ldots, Y_{t-n}\}$includes SP input and output vectors X and Y at several (n) previous time steps. Here we make a reasonable assumption that the unknown (for CAM) imbedded CRM initial conditions ξ_t at the current time step t are a function of SP inputs and outputs at n previous time steps,

$$\xi_t = g(X_{t-1}, \ldots, X_{t-n}, Y_{t-1}, \ldots, Y_{t-n}),$$

which leads to (18).

The equation (18) leads to a more complicated NN emulator (recurrent NN), which will include additional inputs. As a result, the history of the CRM calculations will be taken directly into account when developing the NN emulation. The optimal length of the history needed for NN training (the number of previous time steps, n, to be included), presumably from several hours to 1-2 days (consistent with a cloud life cycle represented by CRM), will be determined experimentally.

The significant speedup provided by NN-MMF, which, in accordance with our estimates, is 250 times faster than the current PNNL-MMF and at least as fast as the current CAM-5, will be used for producing decadal and longer climate runs following, for example, the AMIP protocols. The goal is to determine how well the MMF performs as a climate model compared to observations

and to results from other climate models (e.g. CAM). The current MMF is too computationally intensive to allow even for a single extended run, let alone ensemble runs.

Changes to the current CRM configuration within the MMF are limited largely by a computational overhead. In its 2D configuration at 4 km resolution, the model does a good job of capturing deep convection, hence its improvements are reported in simulations of the tropical Pacific climatology (Ovtchinnikov et al., 2005). However, the 4 km resolution is insufficient to resolve boundary layer clouds adequately (Ackerman et al., 2005; available at science.arm.gov/~ackerman). The ability of the CRM to simulate upper tropospheric cirrus is mixed, with some success in the case of more organized systems, but less so in the case of detached cirrus. The fundamental goal of this and the following works is to develop a fast NN-MMF in order to increase computational throughput. It will allow for improvements in MMF, for example, increasing the horizontal and vertical resolution of imbedded 2-D CRM, improvements of its physics, and eventually substituting the 2-D CRM with the full 3D-CRM. Improvements in the MMF itself through improved CRM physics and coupling will ultimately be incorporated in the NN-SP emulator through re-application of the methodology developed here to the new MMF with 3D-CRM to produce the new NN-MMF, which is computationally feasible.

References

Attali J-G. and G. Pagès (1997) Approximations of Functions by a Multilayer Perceptron: A New Approach, Neural Networks, 6, pp. 1069-1081.

Chen, T. and H. Chen (1995) Approximation Capability to Functions of Several Variables, Nonlinear Functionals and Operators by Radial Basis Function Neural Networks, Neural Networks, 6, pp. 904-910.

Cherkassky V. and F. Mulier (1998), Learning from Data: Concepts, Theory, and Methods, Wiley

Grabowski, W. W., 2001: Coupling cloud processes with the large-scale dynamics using the Cloud-Resolving Convection Parameterization (CRCP). J. Atmos. Sci., v. 58, 978-997.

Grabowski, W. W., 2002: Large-scale organization of moist convection in idealized aquaplanet simulations. Int. J. Numer. Methods in Fluids, 39, 843--853.

Grabowski, W. W., 2003a: MJO-like coherent structures: Sensitivity simulations using the Cloud-Resolving Convection Parameterization (CRCP). J. Atmos. Sci., J. Atmos. Sci., v. 60, 847-864.

Grabowski, W. W., 2003b: Impact of cloud microphysics on convective-radiative quasi-equilibrium revealed by Cloud-Resolving Convection Parameterization (CRCP). J. Climate, v. 16, 3463-3475.

Grabowski, W. W., 2004: An improved framework for superparameterization. J. Atmos. Sci., in press.

Grabowski, W. W., and P. K. Smolarkiewicz, 1999: CRCP: A Cloud Resolving Convection Parameterization for Modeling the Tropical Convecting Atmosphere. Physica D, 133, 171-178.

Khairoutdinov, M. F., and D. A. Randall, 2001: A cloud resolving model as a cloud parameterization in the NCAR Community Climate System Model: Preliminary results. Geophys. Res. Let., 28, 3617-3620.

Khairoutdinov, M. F., and D. A. Randall, 2003: Cloud resolving modeling of the ARM summer 1997 IOP: Model formulation, results, uncertainties, and sensitivities. J. Atmos. Sci., 60, 607–625.

Krasnopolsky V., D.V. Chalikov, and H.L. Tolman (2002) A neural network technique to improve computational efficiency of numerical oceanic models, Ocean Modelling, v. 4, 363-383

Krasnopolsky, V. M., 2007: Neural Network Emulations for Complex Multidimensional Geophysical Mappings: Applications of Neural Network Techniques to Atmospheric and Oceanic Satellite Retrievals and Numerical Modeling, Reviews of Geophysics, 45, RG3009, doi:10.1029/2006RG000200.

Krasnopolsky, V. M., M.S. Fox-Rabinovitz, and A. A. Belochitski, 2008: "Decadal Climate Simulations Using Accurate and Fast Neural Network Emulation of Full, Long- and Short Wave, Radiation.",Monthly Weather Review, 136, 3683–3695, doi: 10.1175/2008MWR2385.1.

Krasnopolsky, V. M., M. S. Fox-Rabinovitz, Y. T. Hou, S. J. Lord, and A. A. Belochitski, 2010: "Accurate and Fast Neural Network Emulations of Model Radiation for the NCEP Coupled Climate Forecast System: Climate Simulations and Seasonal Predictions", Monthly Weather Review, v.138, pp. 1822-1842, DOI: 10.1175/2009MWR3149.1

Krasnopolsky, V. M., M. S. Fox-Rabinovitz, A. A. Belochitski, Philip J.Rasch, P. Blossey, and Y. Kogan, 2011: “Development of neural network convection parameterizations for

climate and NWP models using Cloud Resolving Model simulations", NCEP Office Note 469, http://www.emc.ncep.noaa.gov/officenotes/newernotes/on469.pdf

V. M. Krasnopolsky and Y. Lin, 2012: "A Neural Network Nonlinear Multimodel Ensemble to Improve Precipitation Forecasts over Continental US", *Advances in Meteorology*,Volume 2012, Article ID 649450, 11 pages, doi:10.1155/2012/649450

Krasnopolsky, V., 2013: The Application of Neural Networks in the Earth System Sciences: Neural Network Emulations for Complex Multidimensional Mappings, Atmospheric and Oceanographic Sciences Library, Volume 46, Springer Netherlands; doi:10.1007/978-94-007-6073-8, http://link.springer.com/book/10.1007/978-94-007-6073-8/page/1

Nguyen D, Widrow B (1990) Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. Proc of International Joint Conference of Neural Networks, June 17-21, San Diego, CA, USA, 3: 21-26

Ovtchinnikov, M., T. P. Ackerman, R. T. Marchand, and M. Khairoutdinov, 2005: Evaluation of the multi-scale modeling framework using data from the Atmospheric Radiation Measurement program. *J. Climate*, **19**, 1716–1729, doi: http://dx.doi.org/10.1175/JCLI3699.1

Pincus, R., H. W. Barker, and J.-J. Morcrette, 2003: A fast, flexible, approximate technique for computing radiative transfer in inhomogeneous cloud fields. J. Geophys. Res., Vol. 108, No. D13, 4376, doi:10.1209/2002JD003322

Randall, D., M. Khairoutdinov, A. Arakawa, and W. Grabowski, 2003: Breaking the cloud-parameterization deadlock Bull. Amer. Meteor. Soc., v. 84, 1547-1564.

Ripley, B.D.(1996), Pattern Recognition and Neural Networks, Cambridge Univ. Press.

Taylor, K. E. 2001: Summarizing multiple aspects of model performance in a single diagram, *JGR: Atmosphers*, v. 106, 7183-7192

Wang, M., S. Ghan, R. Easter, M. Ovchinnikov, X. Liu, E. Kassianov, Y. Qian, W. I. Gustafson, V. E. Larson, D. P. Schanen, M. Khairoutdinov, and H. Morrison, 2011: The multi-scale aerosol-climate model PNNL-MMF: model description and evaluation, *Geosci Model Dev*, *4*(1), 137-168.