RC1

This manuscript describes an approach to improve neural network (NN) predictions in settings where the data on which the NN is trained differ (substantially) from those in the setup to which the NN will ultimately be applied. This situation is frequently encountered in earth sciences (and presumably other areas of application), and thus this research topic is highly relevant to applications.

General comment: *The proposed approach to address this issue is more like a general procedure than a specific method, with many modeling choices being left to the user depending on the setup in which this approach is applied. I understand that this is inevitable, but I would then expect a solid theoretical motivation of the proposed approach that gives a good understanding when and why it works. I feel that this is still lacking in the current version of the manuscript. While the three use cases are good examples to demonstrate that the proposed approach can yield notable improvements while also discussing shortcomings and open questions, I must admit that I am somewhat puzzled that the proposed procedure works at all. In the introduction, the authors motivate this research by explaining that in non-linear problems, a systematic shift in the input data does not easily translate to a corresponding shift in the output. But isn't this also true for the NN weights? In my understanding, NN weights are typically not even identifiable, i.e. completely different weights can yield the same (or at least very similar) output. So, how is it possible that that these weights can be extrapolated by (e.g. linear) methods in a stable way? Intuitively, I would expect this to work only if the weights of ParentModeli are guaranteed to remain relatively close to those of ParentModel0, but it doesn't seem that the authors tried to control this in any way. Please provide more motivation and explanation (which could e.g. include a more detailed analysis and visualization of the weight extrapolation underlying the different curved in Fig. 1) of why you expect this approach to work in general, or, which conditions have to be satisfied for this weight extrapolation to work. If any tuning parameters were involved in getting the positive results shown in the three use cases, it would be useful to share this experience to help guide the application of this approach in different settings.*

Response: Thank you for reviewing our paper and for pointing out this fundamental shortcoming. Your understanding is entirely correct in that one needs the ParentModel_i weights to be “close and meaningfully distributed” in relation to ParentModel_0. We added an explanation that states this requirement of the approach. We also explain how to achieve this “close-to-ParentModel_0 ” weight distribution, e.g., by regularization (cf. your last comment). However, there are several ways to achieve such a thing (we mention two). We would like to keep our ways of realizing the approach understood as mere examples. In general, in this paper, we would like to separate “the core approach” and “ways of how to realize its aspects”.

This is what we added to the method section of the manuscript, which has now become the first section after the introduction (we used some of your wording):

“The reasoning why the above approach should work is that fine-tuning the ParentModel_0 further on a training data point x_i will result in a new model (ParentModel_i), whose weights and biases differ a) only slightly and b) in a meaningful way from ParentModel_0. This means that all ParentModel_i form a point cloud around ParentModel_0 in NN parameter space, which encodes the respective sensitivities between this specific ParentModel_0's weights and the variability of the input data. For this to work, the deviations between ParentModel_0 and the related ParentModel_i have to be constrained in a way that no entirely new and therefore unrelated ParentModel_i are created in the process. In other words, all ParentModel_i should be closely yet meaningfully distributed around ParentModel_0.“

We also discuss two ways (as the regularization) that are suitable to achieve this “closeness”. We added:

“To achieve this, it should be theoretically enough to restrict the fine-tuning of ParentModel_0 (cf. Tab. 1, step 2a) towards each x_i to a few update steps (epochs). However, we found that a regularized fine-tuning of ParentModel_0 towards individual training data points x_i works better and allows for more flexible fine-tuning approaches. To regularize the fine-tuning step, we add a representative fraction of the training data to the x_i fine-tuning batch. Naturally, the impact of such a regularization and the impact of the fine-tuning towards a single x_i in generating a new ParentModel_i has to be weighted reasonably. Tab. 1, step 2a gives an example of how to achieve a suitable regularization easily in practice, e.g., we take n random samples from the training data combined with n duplicates of x_i to ensure equal weighting in the fine-tuning batch.”

Consequently, weighting in the given example is 0.5 for each impact, i.e, equal impact from the entire training set ($\mathbb{I}$) and from a single data point x_i.

In addition, as requested by another reviewer, we moved the entire “methods” section to the beginning of the manuscript. In this way, the reader is familiar with the core approach before looking into the examples and the results.

Specific comments:

Comment: *4.1, 2nd paragraph, 'Note, however, that the sensitivity ...': Can you explain this statement further, i.e., in what way do the activation functions affect the sensitivity of the weights? More generally, this entire paragraph seems to assume good familiarity with this type of problem and is somewhat short on details.*

Response: Our reasoning is like this: assume the NN model needs to produce a complicated nonlinear response. If the network were purely linear, during online-learning, respectively, fine-tuning, the weights themselves would need to change in a very complicated way to achieve this. But nonlinear activation functions already introduce curvature and nonlinearity into the system. Because of this, even relatively simple changes in the weights can lead to complex changes in the output. In deep neural networks, this effect becomes even stronger because the work is shared across many layers.

We have now added the following sentences (black) to make this statement (grey) clearer:

“Note, however, that the sensitivity of the weights is not dictated by the domain dynamics alone but also depends on the involved activation functions (and not only the activation function from the node under consideration). For example, if a nonlinear change of model output is required by the application domain dynamics, it can, in principle, be realized by a less complex (e.g., linear) weight change when nonlinear activation functions are involved. Furthermore, the required weight changes can be distributed over several layers, which then will entangle all contributing activation functions.”

Comment: *4.1, 3rd paragraph, 'As there is a clear order ...': This sounds like a substantial departure from the proposed approach. Doesn't this imply that instead of anomalies from ParentModel0, the weight regression is applied to the increments corresponding to subsequent time points. Please clarify.*

Response: The idea here is that the AMOC system changes with time in a continuous way. Ideally, the network weights should also react/change in a continuous way. Resetting the weights to ParentModel0 between every fine-tuning will introduce additional noise into the already noisy process as the fine-tuning randomly navigates to local minima. Without resetting, we smooth the process and hope that the changes in weight space also form a “trajectory” that reflects the trajectory of the system in the AMOC state space.

We have slightly adapted the paragraph. It reads now:

“As there is a clear order of the data in experiment 1, i.e., the temporal order, we skip the "reload ParentModel_0" step of the approach (cf. Tab. 1, step 2c). This way, the online learning is not forgetful, and the weights will evolve continuously along the time axis as the AMOC system does itself. This improves the results in this experiment as it reduces noise in the generation of the weight sensitivities W_ik.

Comment: *4.1, 3rd paragraph, '... every weight and bias of the CNN ...': So, including the weights of the convolution layers? It is again surprising and somewhat counter-intuitive to me that weights of a convolution layer can be (linearly) extrapolated in a meaningful way. Is there any way to visualize the evolution of just the first convolution weights over time, and the associated extrapolation to after the tipping event? Is it possible to quantify in which layer the most impactful extrapolation happens, that improves the NNs performance during and at the tipping event?*

Response: Yes, the convolutional layers, too. Changes in optimal convolutional kernels can be approximated by linear relations at the kernel pixel level. The weight changes, especially in the CNN layers, for sure would be interesting to study as they encode spatial pattern changes of the oceanic convection in this case. Consequently, these would be excellent questions if the scope of the paper were really about investigating the toy problems. Surely your proposed

analyses could be done from the technical point of view (XAI). However, in our manuscript, CNN weights are treated as all other weights, as they do not differ fundamentally from a machine learning perspective. They are learned by back-propagation and should change under changing domain dynamics. To add a respective distinction to the revised paper, just to state that all weights are treated the same way seems a bit unnecessary. But of course, we changed the text for clarification:

"After establishing weight-predictor relations for every weight and bias of the CNN, including the CNN layers, the child model was predicted with sub-models…."

Comment: *4.2, 1st paragraph, '... only individual xi': Can you explain this further, i.e. a) clarify what you mean by individual xi b) what you mean by regularization towards the entire training data set (I believe this has never been explained in detail) and c) why you chose a different approach here.*

Response:
a) As the regularization is now properly introduced in the method section, and the method section is now before all experiments, this statement should make much more sense now. We refer to these "individual x_i" in the following statement from the method section:

"In other words, all ParentModel_i should be closely yet meaningfully distributed around ParentModel_0. To achieve this, it should be theoretically sufficient to restrict the fine-tuning of ParentModel_0 (cf. Tab. 1, step 2a) towards each x_i to a few update steps/epochs only. "

b) You are right, the regularization was not introduced properly. We apologise for the oversight. The reason was that it is not really a part of the weight update framework per se. Details about the regularization were given in the table as a "working example" for how to generate suitable deviations from ParentModel_0 (the ParentModel_i), which stay close to ParentModel_0. Now, we added a new paragraph motivating and discussing the regularization:

"However, we found that a regularized fine-tuning of ParentModel_0 towards individual training data points x_i works better and allows for more flexible fine-tuning approaches. To regularize the fine-tuning step, we add a representative fraction of the training data to the x_i fine-tuning batch. Naturally, the impact of such a regularization and the impact of the fine-tuning towards a single x_i in generating a new \textit{ParentModel_i has to be weighted reasonably. Tab. 1, step 2a gives an example of how to achieve a suitable regularization easily in practice. The weighting in the given example is 0.5 for each impact, i.e, equal impact from the entire training set (I) and from a single data point x_i."

c) No particular reason, we wanted to include another way of keeping the ParentModel_i "close" to ParentModel_0 (see above).

Thank you for your contribution to this manuscript. Your comments helped a lot.

RC2:

Extrapolation has long been recognized as a potentially serious problem when using neural network (NN) models. This manuscript proposes an interesting new method using mainly linear regression to improve on NN extrapolation, which may improve on earlier, simpler linear methods (e.g., <https://www.jeionline.org/index.php?journal=mys&page=article&op=view&path%5B%5D=20 2300493>). There are places in the manuscript where it is hard to follow, and the use of higher order polynomials in the linear regression is a poor approach that should be corrected.

Major comment: -Sect. 4.1, Paragraph 10:
Using polynomials as basis functions in a linear regression model is a bad choice of basis functions when one is extrapolating outside the training data domain. To model a nonlinear relation y = f(x) with a linear regression model, people tend to use a sum of bounded basis functions such as Gaussian or sigmoidal functions, not unbounded basis functions like polynomials, which have wild extrapolation behavior. I think this part needs to be redone with better behaving basis functions.

Response: Thank you for reviewing our paper, your comment, and the link. Your comment for sure contains a lot to consider. In fact, we cite now Hsieh et al. 2023 and some other comparable approaches (e.g., Beucler et al., 2024). As you state yourself, by far the most of the paper depends on linear interpolation (named “polynomials of order 1” in the manuscript) or ELU activation functions, as in the case of the NN-based interpolation in experiment 3 (ELU are bound on one end and linear on the other end). These linear functions and the ELU functions are well-behaved and should not show wild extrapolation behavior outside their fitting range. The polynomials of order 2 are for sure not optimal, as our analysis already shows in the manuscript. They are included to show something different. All these choices are only flavours of implementation of what we call the “core approach”, i.e., the meddling with the internal weights of NN instead of meddling with the input/output data of NN. We want to keep it that way, and the shown regression choices should be understood as examples. We already stated in the initial submission that the choice of regression method should strongly depend on domain knowledge.

Nonetheless, following your suggestion, we tried Gaussian process based NN weight fitting and extrapolation. For all tested cases, the results from the child models generated this way fall between our results of polynomials of orders 1 and 2. However, the fitting of all NN weights (can be easily 100.000 even in a small network) using Gaussian processes is much slower than linear model fits. As a result, having an ensemble run as in experiment 1, we simply can not compute. We think that at least some ensemble with 100 members would be required to include Gaussian processes properly in our paper, i.e., to make it at least somewhat comparable. Even in the case of experiment 2, i.e., where we show only a single realization of child generation, a comparison is hardly possible. The amount of data that Gaussian fitting can manage in a reasonable time is much less than in the linear fitting case. This means we could do the Gaussian fitting only to about 1000 weight sets (1000

ParentModel_i) instead of, e.g., 10.000 as in the linear weight regression case. This way, the performance of the Gaussian fit is strongly limited and hard to compare. However, we are sure that there are use cases for your suggested method (and the many other good extrapolation methods), e.g., when given enough compute or time, or by limiting the NN parameters to fit (cf. experiment 3). Therefore, we added Gaussian processes to the list of regression models in the manuscript. In addition, we now state: “In general, if you know an extrapolation method that works well for your problem, domain or data, then it could also be a good candidate for extrapolating the weights of a NN (if the method is fast enough, that is).”

We also want to mention that the problem of OOD can be less nonlinear within the weight space of a NN than in the out/input space of the data. We state now “... if a complex nonlinear change of model output is required by the application domain dynamics, it could in principle be realized by a less complex (e.g., linear) weight change when nonlinear activation functions are involved.”

Finally, as of request of reviewer #3, we now reference and discuss more regression methods (also Gaussian processes) in general and apply them on a non-NN predictor-output basis. Please see the new Appendix A of the manuscript.

Minor comments:

Comment: *-Sect.2.1, paragraph 3: 1 Sv = 10^6 …*

Response: corrected

Comment: *-Table 1, step 2: What is a "reasonable subset"? 5%, 10%, 20%, 50%?*

Response: Yes, these could all be reasonable choices, depending on the self-similarity the training data has, the size of the network, and the compute at your hands. In the table, we say “for all x_i in I”, i.e., 100%. The same accounts for the application data - if the application data is rather homogenous, one does not need to calculate a single child model for every application data point. We added a corresponding remark to the “Summary and Final Remarks” section.

Comment: *-Table 1, step 2(a): You mean: n copies of a single data point xi ?*

Response: Yes, corrected. We added “a single data point” to the sentence. Side note: This is just the way we thought it would be the easiest to write it (the regularization) down. Surely, within a specific training batch, you do not have to recalculate the gradients of identical copies of single xi as they would always give the same gradient (the gradients differ only between batches). Consequently, we could also reformulate this step to “for each batch: calculate the gradient w.r.t. x_i once, and calculate a weighted average with the gradients that correspond

to n x_j (which are not x_i)". This would be computationally more efficient but harder to write down and script. We would like to keep our shorter way of noting this down.

Comment: *Also, when choosing the xi data points, would you try to choose ones that are closer to the outer boundary of the data cloud than near the center of the data cloud, which presumably would not help with the extrapolation problem?*

Response: This is a really good suggestion. We did not think about this until now. This could pose one of the many hoped-for and needed improvements to this approach. We will try this out in the next study.

Comment: *-Table 1, step 3: The word "target data" is confusing, since in standard NN terminology, in (x,y) regression problems, the target data are the observed values of y. Here, it seems to mean data points in the training data set?*

Response: True, "target" is used in several different ways in our manuscript; this needs fixing. We now speak of targets only in the sense you mentioned, and changed the previous "target" to application data/region when we mean application of a trained NN to data.

Comment: *-Sect. 4.1, paragraph 2: EOFs are now a little old-fashioned in the age of NN. For future research, one could use an autoencoder NN with the neurons in the bottleneck layer giving the nonlinear principal components (the NN can either be the multi-layer perceptron model or the CNN model). With CNN, one could use a U-Net model to extract nonlinear principal components (<https://gmd.copernicus.org/articles/13/1609/2020/>) (e.g. have 4 neurons in the bottleneck layer to give 4 nonlinear principal components).*

Response: Again, this is a really good suggestion. Thank you for this. However, this also does not fall within the general approach as we understand and present it. We added a sentence to the paper:

"At its core, our approach applies existing extrapolation techniques to the inner weights of a NN instead of applying them to the in- and output data as other approaches suggest (e.g., Hsieh, 2023; Beucler et al., 2024)".

Your very good suggestions (also the one above) fall under "(improved) ways of implementing the core approach". We aim to keep the core approach and implementation separated throughout the paper. The given experiments themselves and the many choices we discuss, as well as the choice to use EOF for experiment 1, are mere examples of the implementation of the core approach. Still, it is a good idea, and we might use it soon.

Comment: *-Sect. 4.1, Paragraph 10: "regressions of polynomials of order 1 and order 2 give comparable results" does not look right. I think order 2 is noticeably worse than order 1, e.g.*

in the box plots (Fig. 2), the bottom of the blue boxes went below the zero line for order 2 but not for order 1.

Response: True & Corrected. We changed the sentence to “Since the regressions of polynomials of order 1 and order 2 give comparable results (with a slight advantage of the order 1 results), this ...”

Comment: *-Sect. 4.3, paragraph 2: ELU activation functions: Are the ELU functions used in both layers?*

Response: Yes. We added, “All layers use ELU activation functions...”

Comment: *-Sect. 4.3, final paragraph: I don't see anything contradictory. A single NN model is developed to do interpolation, not extrapolation. You then develop a second NN to help with extrapolation.*

Response: Our reasoning here was that the helper-NN is also trained only on the training data and then is applied to OOD data - OOD to ParentModel_0 as well as OOD to the helper-NN. That this still works is indeed slightly contradictory to us. We think it shows that if a NN is trained on a nonlinear problem, then the problem can become less nonlinear in the weight-space of the NN. Linear weight changes would still give nonlinear output changes when nonlinear activation functions are involved. We have rewritten this paragraph to reflect the above thinking more clearly:

“That a NN can be used to improve the OOD problem of another NN seems contradictory at first. Overall, the terrestrial input is equally OOD to the \textit{ParentModel$_0$} as well as to the child-generating NN.

One could ask why the child-generating NN is not limited by OOD, and why then, not one NN alone can handle the problem. A full answer to these questions remains open to further research. So far, we can only argue that we, as users, know if an OOD task we gave to a NN remains generally the same but has to be adapted to perform outside the training realm. Consequently, splitting an OOD experiment into problem-solving and problem-adaptation represents a form of structural implementation of prior knowledge. This understanding, the required task separation, and the subsequent training of both NNs, a single NN cannot (yet) develop itself. Furthermore, as already discussed, a nonlinear domain shift in an NN's in- and output data could be less nonlinear in the weight-space of the same NN when nonlinear activation functions are involved.”

Thank you for your help with our manuscript. We think it the paper has become much clearer now.

RC3:
The manuscript, "Conditional updates of neural network weights for increased out of training performance," presents a novel methodology aimed at improving neural network (NN) performance when the training data are not representative of the distribution encountered at inference time. This includes both out-of-distribution scenarios and shifts between application regimes. The method is evaluated in three different settings, where the adapted models are reported to outperform the corresponding baseline model.

General comment:

For all three applications, the proposed method is compared only against a "naive" baseline model. While the authors demonstrate improvements relative to this baseline, the absence of comparisons with alternative approaches developed for related out-of-distribution or regime-shift problems makes it difficult to assess the practical significance of the reported gains. The challenge of degraded deep-learning performance under distribution shift has been extensively studied, yet the manuscript does not provide any comparison with existing methods from that literature. Although the proposed framework may be more general than many previously published approaches, the current results lack a meaningful frame of reference beyond a standard neural network. I therefore encourage the authors to include, for each application, at least one additional benchmark method, training strategy, or architecture from the relevant literature. This would allow the reader to assess whether the proposed approach offers advantages in predictive skill, computational cost, robustness, or generality relative to existing alternatives. If the authors do not wish to benchmark against a broad range of methods, they should at least justify why the chosen baseline is sufficient for each application and clarify what specific advantage their method is intended to provide over existing approaches.

Response: Thank you very much for reviewing our paper, and for this general comment, as we think it might point to an unclear definition of the scope of the paper. We make it now clearer at the manuscript beginning what this paper is about and what we think it should not be about:

"However, this paper does not discuss if and when NN are suitable choices for nonlinear OOD problems. Furthermore, we will not discuss (in depth) the many other methods that might be able to extrapolate the inputs or outputs of some NN to OOD regions. The reason is that many tasks do require NN, and these other approaches cannot be used, e.g., reasoning and decision making, (conditional) generative AI, big data, and large classification problems such as image feature detection. These are applications where the extrapolation or averaging (cf. Hsieh, 2023) of NN output will not be meaningful. To conclude, this study aims at problems where NNs are used for various reasons, e.g., ease of use and implementation, existing workflows, compute limitations, invertibility, lack of alternatives, etc., and these NNs should be used on OOD data. Still, for completeness, we provide a comparison of our experiment 1 results to several standard extrapolation methods in Appendix A, together with a brief discussion."

In this light, the “naive” baseline model we chose, i.e., the parent model, is the only one that really matters. In the end, we aim on purpose to extrapolate the operator, not the input, nor the output. We are sure this is useful for many scientists dealing with NN. In this view, the only other baseline approach we see in the literature is that from Hsieh (2023) – e.g., averaging NN ensemble output. However, this is essentially what we do in the boxplots (Figs. 2&4) with the difference that we use NN models of the same configuration for the ensemble average, while Hsieh also discusses ensemble members that differ in their activation functions. Furthermore, there is one interesting approach we did not try that is linear extrapolation of NN output at the boundary of the training data distribution towards the OOD region (Hsieh, 2023). Going from the 1D example to higher dimensions is not straightforward. The methods discussed by Hsieh apply mostly to low-dimensional and foremost continuous numerical output. Our method would work with classification and generative AI, too, where a NN’s output cannot be meaningfully averaged or extrapolated. We therefore added:

“These are applications where the extrapolation or averaging (cf. Hsieh, 2023) of NN output will not be meaningful."

Still, to neglect the comment, as probably other readers might have the same question. We did calculate the performance of several additional baseline methods operating only on the input and output data level: Gaussian processes, support vector machines (w. linear and radial kernels), random forest, linear regression (orders 1 and 2). We added an appendix (Appendix A) dealing with these baseline approaches. Still, we would like to mention that a) some of these approaches might struggle with more complex tasks and very high dimensional input/output, b) all these methods might also be a good choice as regression models to work within our approach, i.e., use them to predict the weights of the child models, c) given the scope of the paper, it is not really clear what it would mean if some of the standard methods would really outperform our approach (what they do not do) in these simple toy models. As already stated, these toy problems do not pose any problem to standard physics/math.

Having said this, one can see in the new plots of the appendix that a) even our previous “naive” baseline, i.e., ParentModel_0, outperforms most of the newly added baseline models, and is quite comparable to the best, b) the ChildModels outperform all of them, and c) the degree to which the newly added baseline models perform depends on the problem at hand.

Overall, we added 3 plots for the mentioned baseline models to the manuscript’s new appendix. The corresponding text reads:

“To provide a baseline, we present and compare a range of standard linear and nonlinear methods used to extrapolate the targets of experiments 1 and 2, i.e., AMOC strength and sea water density, without any use of NN. The methods (Gaussian process, linear model, random forest, and support vector machine) were fitted to the same predictors as the weight prediction method, i.e., 4 leading EOF (experiment 1) and S, T, P, φ, λ, and z (experiment 2). Likewise, the respective OOD projections are based on the same predictors that our weight prediction method uses. Please note that none of these methods outperform the child models in the OOD application cases. Most of the methods do not outperform the parent NN

(ParentModel_0) either. However, even if a method exists that outperforms parent and child NN in our toy OOD problems, it is not clear what we can learn from this, as the results surely depend on the complexity of the task at hand. However, our approach remains useful in all cases where NNs are used for one of the various reasons, e.g., ease of use and implementation, existing NN workflows, compute limitations, invertibility requirements, or simple lack of alternatives (e.g., decision making, generative AI, image classification)."

Minor comments:

Comment: *The proposed approach involves several methodological choices, including, for example, the selection of weight-prediction and weight-regression strategies. A summary table for each experiment, listing the specific design choices adopted, would help the reader follow the setup and focus more clearly on the results.*

Response: Good suggestion, we added small tables to the results section in front of each experiment. Here, we now state the deviations from the implementation examples given for the core approach in Tab.1.

Comment: *The first paragraph in Section 1 provides well cited statements. However, the remainder of the introduction has many statements which are not supported by citation, even though they would require them.*

Response: We added more citations to the introduction and other parts of the manuscript as well. In addition, we are thankful for more literature suggestions.

Comment: *Some acronyms are introduced without definition, for example AI and ARGO. All acronyms should be defined at first use.*

Response: Corrected. Argo historically stopped being an acronym. We changed the writing from ARGO to Argo. We added more acronym definitions, too.

Comment: *There are some inconsistencies in acronym formatting and capitalization. For example, neural networks is not capitalized when the acronym is introduced in Section 1, whereas Equation of State is capitalized when introduced in Section 2.2. The authors should check the manuscript for consistency in this regard.*

Response: We aim for lower case unless the acronym refers to a proper name. However, what is a proper name or not can surely be debated (e.g., Equation of State).

Comment: *In Section 2, I would recommend presenting the methodology before the individual experiments. This would allow the reader to understand the general framework before encountering the application-specific implementations.*

Response: Very good point. We moved the methodology section to before the experiments, i.e., right after the introduction.

Comment: *In Section 2.3, the terminology may need to be reconsidered. I am not convinced that cross-domain is the most appropriate description of Experiment 3. Based on both the setup and the later discussion, this case seems more naturally characterized as spatiotemporal extrapolation.*

Response: The motivation here was that the dynamics over land are very different than over the ocean. But yes, the dynamics do also share some common ground. We changed the wording to “cross-regime”. We did change domain → regime also in other similar usages throughout the paper.

Comment: *There are a number of grammatical issues and typographical errors throughout the draft. For example, phrases such as “the training bases only on data” and “Polynomials of order 1 and 2 where fitted” should be corrected.*

Response: Thank you, corrected. We did our best to further improve the language throughout the entire manuscript.

Comment: *I would encourage the authors to provide access to an implementation of the proposed approach, for example through a public code repository. This would greatly facilitate reproducibility and allow readers to test and use the method more easily.*

Response: A very good and valid request. We cleaned one of our scripts for experiment 1 of all unnecessary ballast and uploaded it. We will make it publicly available and link it in the acknowledgements in case of final publication. As this rebuttal is public, we, for now, would like to share the link only with the reviewers. We will do this via the editor.

Thank you for your help with our manuscript. It really helped to clarify the scope of the manuscript considerably.